



**Konzept und Ansatz einer Wertschöpfungskette für die Erkennung und
Bereitstellung neuer Fahrumfänge intelligenter Fahrzeuge**
Bachelorarbeit

An der
Carl von Ossietzky Universität Oldenburg
Studiengang Wirtschaftsinformatik

Vorgelegt von Linus Hestermeyer
linus.hestermeyer@gmail.com
Matr.Nr.: 4087097

Erstprüfer: Prof. Dr. Frank Köster
Zweitprüfer: Dipl.-Inform. Gerald Sauter

Oldenburg, den June 4, 2020

Vorwort

Diese Arbeit entstand im Rahmen meines Wirtschaftsinformatik Studiums an der Carl von Ossietzky Universität Oldenburg.

Mein Besonderer Dank gilt meinen Betreuern Gerald Sauter und Prof. Dr. Frank Köster für die stetige fachliche Betreuung und Hilfe bei der Erstellung der Arbeit und des Prototypen. Der Gedankenaustausch über die Thematik und die Änderung des Kernthemas der Arbeit haben das Ergebnis erst möglich gemacht.

Selbstverständlich gilt mein Dank auch meinen Eltern und Geschwistern Lukas und Johanna, die mich bei der Entwicklung meiner fachlichen und beruflichen Laufbahn stets unterstützt haben.

Nicht zuletzt möchte ich meiner Freundin Julia, der Fachschaft Informatik der Universität Oldenburg und dem OFFIS danken, welche während des Studiums immer eine gute Anlaufstelle für fachliche aber auch persönliche Aspekte meiner selbst gewesen sind.

Oldenburg, im Juni 2020

Linus Hestermeyer

Inhaltsverzeichnis

1 Motivation	1
2 Theoretischer Rahmen	2
2.1 Einführung	2
2.2 Bedarfserkennung von Software	3
2.2.1 Relevante Systeme eines autonomen Fahrzeugs in der Bedarfserkennung	3
2.2.2 Eine geeignete Kommunikationsgrundlage: OpenSCENARIO	4
2.2.3 Software-User-Pattern-Recognizer [SUPR] im Rahmen der Bedarfserkennung	6
2.3 Bereitstellung von Software	8
2.3.1 OTA-Aktualisierungen mit UPTANE	8
2.3.2 Anpassung von UPTANE Architektur	9
2.3.3 SUPR im Rahmen der Softwarebereitstellung	11
2.3.4 Die Mensch-Maschine-Schnittstelle	15
2.4 Technologie- und Methodik-Scouting	16
2.4.1 Arbeitsmethodik: User-Centered-Design	16
2.4.2 Tech-Scouting	19
2.5 Ausblick und erstes Konzept der praktischen Umsetzung	20
3 Das Business Model Canvas und Wertschöpfungskette	24
3.1 Business Model Canvas	24
3.1.1 Kundensegmente	25
3.1.2 Kundenbeziehungen	26
3.1.3 Marketingkanäle	26
3.1.4 Nutzenversprechen	27
3.1.5 Einnahmequellen	28
3.1.6 Schlüsselressourcen	28
3.1.7 Schlüsselaktivitäten	28
3.1.8 Schlüsselpartner	29
3.1.9 Kostenstruktur	30
3.2 Konzept einer Wertschöpfungskette	31
3.2.1 Primäre Aktivitäten	31
3.2.2 Unterstützende Aktivitäten	37
3.3 Zusammenfassung	38
4 Technische Konzepte	39
4.1 Klassifizierung von Software	39
4.1.1 Berechtigungen und Details	39
4.1.2 Softwarekennzahlen	41
4.2 Kommunikationsprotokoll	42
4.2.1 Kommunikationsprotokoll: Selbstständiger Softwarekauf	42
4.2.2 Kommunikationsprotokoll: Installationsvorschlag von Service Provider	43

4.2.3	Kommunikationsprotokoll: Nutzung eines Service	44
4.3	Ausblick	47
5	Vorstellung des Prototypen	47
5.1	Architektur des Prototypen	47
5.1.1	Architektur der Software Applikation Managers (SAM)	49
5.1.2	Architektur des OEM-Servers (Server)	50
5.2	Installation	50
5.2.1	Server und SAM Installation	51
5.2.2	MMS Installation	51
5.2.3	Carla Installation	52
5.3	Nutzung des Prototypen	53
5.4	Analyse des Prototypen und Ausblick auf die Weiterentwicklung	54
6	Resumé	56
7	Ausblick	56
7.1	Konzept Individuelle Darstellung von Softwares im Shop	56

1 Motivation

Wird ein Blick in die nahe Zukunft der Automobilindustrie geworfen eröffnet sich ein breites Feld neuartiger Technologien. Autonomes fahren, elektro-betriebene Motoren, C2X-Kommunikation, Car-Sharing und weiteres versprechen "eine Zukunft, in welcher sich unser Verständnis des Automobils grundlegend ändern wird."¹ Durch die neuen Technologien kommt es zu Verschiebungen in den Absatzkanälen von Automobilherstellern. Der klassische Verkauf von Neuwagen wird zurückgehen und durch neu entstehende Seitenmärkte, wie beispielsweise dem Erkennen und Bereitstellen neuer Fahrumfänge verdrängt.²

Aktuell sind Fahrzeuge bereits dazu in der Lage, gewisse Verkehrssituationen wie Einparken oder das Fahren im Stau selbstständig zu bewältigen. In absehbarer Zeit wird sich das Spektrum dieser Applikationen vergrößern und Fahrzeughalter sollen dazu in der Lage sein, diese neuen Applikationen auf ihrem Fahrzeug zu installieren, um somit dessen Fahrumfang stetig zu erweitern. Fahrzeughalter sollen bei der Entscheidung unterstützt werden, welche Applikationen sie tatsächlich benötigen, damit Kosten als auch die Ressourcen des Fahrzeugs zu sparen.

Um auf diese Änderungen vorzubereiten, wird im Rahmen dieser Arbeit ein Konzept einer Wertschöpfungskette aus der Sicht eines Automobilherstellers erarbeitet. Darüber hinaus wird eine Simulation entwickelt, welche im Kontext automatischer Parkplatzfindung einen Absatzprozess von Software darstellt. Die Entwicklung dessen orientiert sich an dem UPTANE-Standard³, welcher im Hinblick auf eine sichere Kommunikation zwischen Fahrzeugen und Servern entwickelt wurde.

¹Herbert Dies

²ericcson

³uptane

Das folgende Kapitel während des vorangegangenen Forschungsseminars erstellt.

2 Theoretischer Rahmen

2.1 Einführung

"Durch den Einsatz von Elektrik und Elektronik ist das Automobil in den vergangenen Jahrzehnten stark geprägt worden, und die „Intelligenz“ in den Subsystemen hat exponentiell zugenommen." (Vgl. [?, S. 1]) Mit zunehmend mehr intelligenten Fahrassistenten in modernen Fahrzeugen ist es Absehbar, dass bereits in naher Zukunft Fahrzeuge die Stufe 3 bzw. teilweise Stufe 4 des Autonomen Fahrens erreichen. Ein Fahrzeug der Stufe 3 ist dazu in der Lage dazu, die Längs- und Querlenkung in bestimmten Anwendungsfällen selber übernehmen und diese so sicher zu durchfahren. Hierbei wäre allerdings noch ein Insasse notwendig, der in einem Notfall das Steuer übernehmen kann. In Stufe 4 Fahrzeugen kann das Fahrzeug die komplette Fahraufgabe in bestimmten Anwendungsfällen übernehmen. [?, S. 14]

Erst Stufe 5 Fahrzeuge werden "vollumfänglich auf allen Straßentypen, in allen Geschwindigkeitsbereichen und unter allen Umfeldbedingungen die Fahraufgabe vollständig allein durchführen. Wann dieser Automatisierungsgrad erreicht sein wird, kann heute noch nicht benannt werden." (Vgl. [?, S. 14]). Aufgrund dessen ist es wichtig, dass Stufe 3 und Stufe 4 Fahrzeuge in Zukunft mehr Anwendungsfälle abdecken können. Hierzu sollen diese über eine kabellose Schnittstelle orts- und zeitunabhängig Softwarepakete herunterladen können, welche das Spektrum der autonomem Fahrfunktionen des Fahrzeugs zweckgebunden erweitert. Angenommen Sie planen mit ihrem neuen Fahrzeug eine Autofahrt nach England. Spätestens ab dem Ende des Eurotunnels wäre es für das Fahrzeug nicht mehr möglich selbstständig zu fahren, da dort Linksverkehr herrscht. Das Auto soll automatisch erkennen können, dass es ab einem bestimmten Punkt nicht mehr selbstständig fahren kann. In Folge dessen soll es eine Software zum Kauf/Miete anbieten, welche es dem Auto ermöglicht, am Linksverkehr teilnehmen zu können. Die Halter können ihr Fahrzeug dementsprechend zunehmend autonom fahren lassen, was den Marktwert des Autos nach dem Kauf steigern kann. Die Fahrzeughalter sollen bei Kauf von Software vom System unterstützt werden in Form von Vorschlägen für neue Software, die den Bedarf des Fahrers abdeckt. Diese Vorschläge sollen zu Zeitpunkten erfolgen, in denen der Verkauf einer bestimmten Software möglichst wahrscheinlich ist. Mittels dessen wird zudem unterbunden, dass der Nutzer von zu vielen Benachrichtigungen überfordert wird.

Damit Anwendungsfälle rasch abgedeckt werden können, bedarf es einem großen Spektrum an Software, die eben diese abdeckt. Um dies schnellstmöglich bewerkstelligen zu können ist es erforderlich, dass die Entwicklung von Softwarepakete durch Zulieferer geschehen, welche sich explizit auf diesen Markt fokussieren. Der hierdurch entstehende Seitenmarkt für die Entwicklung von Fahrtfunktionssoftware kann somit schnell wachsen und sich als Teil in der Automobilindustrie etablieren. Durch autonome Fahrfunktionen werden Autos in der Regel schonender gefahren als vom Menschen, weshalb die Lebenszeit von Autos voraussichtlich verlängert wird. [?, (S. 16)] Ericssons Juergen Daunis sagt diesbezüglich, dass die meisten Analytiker und Führungskräfte der Meinung sind, dass der Umsatz dieser neu entstehenden Seitenmärkte weiter steigen wird und dass das traditionelle Geschäftsmodell an dem wirtschaftlichen Maximum seiner Existenz ist. [?]

Die in dieser Arbeit erfassten Gliederungspunkte dienen als Grundlage für die gleichnamige Bachelorarbeit und sind im Wesentlichen als Literaturrecherche und Grundlagenerarbeitung zu verstehen. Es werden einzelne Bausteine der in der Bachelorarbeit zu erstellenden Wertschöpfungskette erstmalig benannt und konkretisiert. In Kapitel zwei wird dargestellt, wie das Auto einen Softwarebedarf erkennt, wie es einen Server hierüber informiert und wie dieser Server die richtige Software sucht. In Kapitel Drei wird zum einen eine sichere Architektur für kabellose Aktualisierungen erarbeitet. Des weiteren wird erläutert, wie Software verkauft wird und es werden Richtlinien für die Mensch-Maschine-Schnittstelle erarbeitet, auf welcher ein Angebot letzten Endes angezeigt werden soll.

2.2 Bedarfserkennung von Software

Im Rahmen der Bedarfserkennung steht die Beschreibung der eigenen Umgebung von Fahrzeugen im Mittelpunkt. Damit die Softwarehersteller den Bedarf von Fahrzeugen abdecken können, muss ihnen die Umgebung des Fahrzeugs zu dem Zeitpunkt der Bedarfserkennung bekannt sein. Um zu verstehen, wie ein Fahrzeug seine Umwelt eigens beschreiben kann, wird im Folgenden die Architektur eines autonomen Fahrzeugs vorgestellt und erläutert, welche Bestandteile dessen für die Bedarfserkennung relevant sind.

2.2.1 Relevante Systeme eines autonomen Fahrzeugs in der Bedarfserkennung

Um die Suche nach Software zu erleichtern, benötigt die Suche auf dem Server gewisse Daten als Input. Damit die Situation, in welcher das Fahrzeug nicht selbstständig fahren kann, durch eine Software abgedeckt werden kann ist es wichtig eine Beschreibung der Umwelt des Autos zu dem Zeitpunkt der Bedarfserkennung zu erstellen. Durch die Fusion aufgenommener Kamera-, Ultraschall- und Radardaten kann die Umwelt beziehungsweise die Umgebung des Autos in einem Datenformat wie dem von der ASAM.¹ definierten Standard "OpenSCENARIO" [?] dargestellt werden. Dieser wird in Kapitel 2.2.2 vorgestellt.

Im Rahmen der Suche nach möglicherweise nötiger Software spielt die Routen- und Bewegungsplanung des Autos eine große Rolle. Zum einen kann das Auto bei der Eingabe einer neuen Route diese nach Gegenden absuchen in denen oft neue Software benötigt wird. Es kann infolgedessen diese dem Fahrer bereits vor Fahrtbeginn vorschlagen und somit die Bequemlichkeit der Fahrt sicherstellen. Zum anderen soll das Auto Muster im Fahrtenverlauf erkennen, um so bei der Erkennung eines Softwarebedarfs auf einer dieser Strecken eben diese Software zum Kauf vorzuschlagen.

Abbildung 1 zeigt die Architektur eines Autonomen Fahrzeugs nach Jeff Schneider. Sie verdeutlicht, welche Bestandteile ein Autonomes Fahrzeug besitzt um mit Hilfe dieser selbstständig zu Fahren. Die aufgenommenen *Sensordaten* leiten Informationen an die *Karten- & Positionsverfolgung* weiter. Durch einen Merge (Zusammenführung) dieser Daten kann das Fahrzeug die eigene Umwelt *wahrnehmen* und mit Hilfe der dynamischen Inhalte der Welt (zB. Geschwindigkeiten) *Vorhersagen* für den Verkehr stellen. Als Bündel stellen sie die Bewegungsplanung dar.

¹<https://www.asam.net/standards/detail/openscenario/>

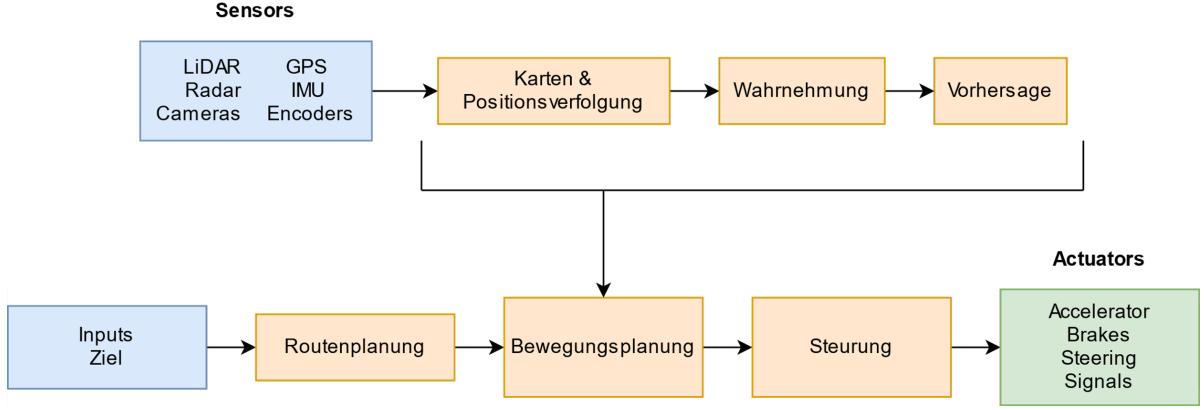


Abbildung 1: Architektur AV nach Jeff Schneider (CMU) [?]

Der Fahrer gibt dem Auto initial ein *Ziel*, mit Hilfe dessen das Fahrzeug die zu fahrende Route berechnet. Die Bewegungssteuerung gibt das Wahrgenommene und Vorhergesagte weiter an die Steuerung welche letztendlich entscheidet, was die einzelnen Aktuatoren machen. Damit der Bedarf einer Softwarelücke festgestellt werden kann, müssen die Systembausteine "Sensoren", "Karten & Positionsverfolgung", "Wahrnehmung" und "Vorhersage" angeknüpft werden um so herauszufinden **wann** das Fahrzeug nicht mehr selbstständig fahren kann. Wurde dieser Moment festgestellt, wird die Übergabe der Fahraufgabe initiiert und zeitgleich auch die Suche nach Software gestartet.

Neben den wichtigen Bausteinen der Bewegungsplanung kann auch anhand der Routenplanung von Fahrzeugen ein Softwarebedarf untersucht werden (Siehe 2.2.3). Zunächst wird eine Möglichkeit dargestellt, wie die Daten der Bewegungsplanung in einer geeigneten Form gespeichert und versendet werden können.

2.2.2 Eine geeignete Kommunikationsgrundlage: OpenSCENARIO

Der Inhalt dieses Kapitels basiert auf Inhalten der Projektwebseiten des OpenSCENARIO Standards. [?]. OpenSCENARIO ist ein XML-basiertes Dateiformat zur Beschreibung aller statischen (*Gegenstände*, *Hindernisse*, etc.) und dynamischen (*Geschwindigkeiten*, *Bewegungsrichtungen*, etc.) Inhalte der Umwelt eines Fahrzeugs. "Der primäre Anwendungsfall von OpenSCENARIO ist es komplexe, synchrone Manöver zu beschreiben, welche mehrere Entitäten wie Fahrzeuge, Fußgänger und andere Verkehrsteilnehmer betreffen." (Vgl. [?, asam.net])

Abbildung 2 zeigt einen Ausschnitt einer OpenSCENARIO-Datei. Für ein Szenario ist immer das **Straßenetzwerk** (*RoadNetwork*) sowie die beinhalteten **Entitäten** (*Entities*) festzulegen. Das eigentliche Szenario ist innerhalb eines **Storyboards** dargestellt und unterteilt sich in einzelne **Storys** (Siehe Abbildung 2). Alle zuvor definierten Entitäten erhalten eine initiale **Action**, welche das initiale Handeln der Entität festlegt (Siehe *<Init>-Block*).

Eine einzelne Story ist immer einer einzelnen Entität zuzuordnen. Eine Story wiederum ist in "**Acts**" aufgeteilt, welcher eine Sammlung an "**Sequenzen**" beinhalten kann. Jedes dieser Elemente kann mit einer "**Condition**" versehen werden. Wird die "Condition" (Bedingung) erfüllt, wird der jeweilige Story-/Act- oder Sequenz-Block ausgeführt.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <OpenSCENARIO>
3
4  <FileHeader revMajor="0" revMinor="9" date="2017-02-24T10:00:00"
5    description="Sample Scenario - Overtaker" author="Andreas Biehn"/>
6
7  <ParameterDeclaration/>
8
9  <Catalogs>
38
39  <RoadNetwork>
40    <Logics filepath="Databases/PEGASUS/PEGASUS_A01.xodr"/>
41    <SceneGraph filepath="Databases/PEGASUS/PEGASUS_A01.opt.osgb"/>
42  </RoadNetwork>
43
44  <Entities>
45    <Object name="Ego">
51    <Object name="A1">
57  </Entities>
58
59  <Storyboard>
60    <Init>
61      <Actions>
62        <Private object="Ego">
79        <Private object="A1">
96      </Actions>
97    </Init>
98    <Story name="MyStory" owner="A1">
99      <Act name="MyAct">
100        <Sequence name="MySequence" numberOfExecutions="1">
175        <Conditions>
186        </Act>
187      </Sequence>
188    </Act>
189  </End>
190  </End>
191 </Storyboard>
192 </OpenSCENARIO>

```

Abbildung 2: Ausschnitt einer OpenSCENARIO Datei von Andreas Biehn [?, (Downloads)]

OpenScenario ist aus mehreren Gründen gut geeignet um als Kommunikationsgrundlage zwischen Auto und Server zu fungieren. Zum einen handelt es sich dabei um eine Opensourcelösung, wodurch jeder Entwickler Weltweit selbstständig mit diesem Arbeiten kann. Zum anderen existiert wegen dem XML-basierten Dateiformat eine gute Lesbarkeit und Regelmäßigkeit Abfolge der Szenarien. Ein Szenario kann aufgrund

dieser Regelmäßigkeit einfach und schnell erstellt werden. Der letzte und größte Vorteil von OpenScenario ist es, dass man ein Szenario in dem Opensource Simulator "Carla" abspielen kann.

Der im Folgenden vorgestellte Software-User-Pattern-Recognizer (SUPR) nutzt das OpenScenario-format zur Suche nach Software. Welche Suchvarianten es gibt und weitere Aufgaben des SUPR werden hier vorgestellt.

2.2.3 Software-User-Pattern-Recognizer [SUPR] im Rahmen der Bedarfserkennung

Das in diesem Abschnitt vorgestellte Konzept des Software-User-Pattern-Recognizer (kurz: SUPR) ist ein eigens ausgearbeiteter Baustein, welcher die Bedarfserkennung für intelligente Fahrzeuge beschleunigen und vereinfachen soll. Die Aufgaben des SUPR lassen sich in Aspekte der Bedarfserkennung sowie der Bereitstellung aufteilen, weshalb der SUPR in Kapitel 3.3 wieder aufgegriffen wird.

Eine Aufgabe des SUPR im Rahmen der Bedarfserkennung ist es, auf Anfrage hin eine Suche nach passenden Softwarepaketen durchzuführen (1.). Die zweite Aufgabe ist die regelmäßige Suche nach Software in der Umwelt/Region des Autos bzw. die Suche nach Softwarepaketen entlang zu Fahrender Strecken (2.). Das Maß der Rechenleistung auf Seiten des Servers ist indes höher als auf dem Fahrzeug. Eine mögliche Gegenüberstellung von Rechenleistung und Sendeleistung des Servers und des Autos kann zu behebende Defizite der Architektur aufdecken - die Optimierung des Systems ist jedoch nicht Teil des Forschungsseminars. Der Server, auf welchem die Suche stattfindet, benötigt zwei Datenbanken: die eine enthält sämtliche Softwarepakete für intelligente Fahrzeuge, die andere eine große Menge an OpenScenario Dateien, welche zusätzlich Fremdschlüsselverweise auf ein oder mehrere Softwarepakete bereitstellen. Nur wenn diese Bedingung erfüllt ist, kann eine Suche durchgeführt werden.

1. Gezielte Suchanfragen

Gerät ein intelligentes Fahrzeug im Laufe seiner Fahrt in eine ihm unbekannte Situation, kann es diese mit Hilfe von OpenSCENARIO beschreiben. Das vom Auto erstellte Szenario wird an den Server geschickt, welcher dieses mit den auf der Datenbank liegenden Szenarien abgleicht. Ist die Suche abgeschlossen, wird eine Fallunterscheidung zwischen den folgenden Optionen getätigert:

A Es wird kein passendes Softwarepaket gefunden

Entweder existiert zu dem übergebenen Szenario keine Software oder möglicherweise wurde die im Szenario dargestellte Situation noch nicht zu einer Software gemapped. Dies sollte von speziell hierfür angestellten Arbeitnehmern überprüft werden, um so Dopplungen in der Softwareentwicklung zu vermeiden.

B Es werden (mehrere) passende Softwarepakete gefunden.

In diesem Fall gilt zu entschieden, welches Softwarepaket die besten Auswirkungen auf die Performance des Autos hat. Hierzu ist das Heranziehen diverser Kennzahlen ratsam um somit die Performance der unterschiedlichen Softwares untereinander zu vergleichen und eine fundierte Entscheidung treffen zu können. Eine beispielhafte Ausarbeitung dieser Entscheidungstreffung stellt Niklas Stelter in seiner Bachelorarbeit vor [?]. Am Ende einer Suche soll entweder eine einzelne oder keine Software vorgeschlagen werden.

Neben der vom Fahrzeug ausgehenden Suche nach einer bestimmten Software, kann dieses auch Vorschläge für Software von einem Server erhalten, welcher dauerhaft die Umwelt intelligenter Fahrzeuge analysiert.

2. Ortsbezogene Erkennung eines Softwarebedarfs Neben der Suche nach einer bestimmten Software soll der SUPR auch Suchen in der Heimatregion des Autos durchführen sowie auf zu Fahrenden Strecken. Diese Suchen basieren nicht auf OpenScenario-Dateien sondern auf Basis der Routenplanung[A] (Siehe Abbildung 1) und der Fahrtenhistorie des Fahrzeugs[B].

A: Suche auf Basis der Routenplanung

Wird vom Fahrer eine nicht oft oder noch gar nicht zurückgelegte Strecke vorgegeben, so soll der SUPR entlang der zu fahrenden Strecke häufig auftretenden Softwarebedarf identifizieren und dem Fahrer vor oder kurz nach Antritt der Reise diese Softwarevorschlägen. Daraus ergibt sich die Anforderung, zu jedem sich in der Datenbank befindlichen OpenScenario auch die geographische Position der Bedarfsentdeckung zu speichern. In Abbildung 3 ist folgende Situation dargestellt: Unser Auto hat als Ziel im Navigationssystem "Braunschweig" erhalten und befindet sich aktuell auf der A2. Die roten Boxen stellen den möglichen Bereich dar, in welchem der Server die Umweltanalyse betreibt. Wird eine Software entdeckt, überprüft das System die Relevanz für das eigene Fahrzeug und schlägt sie je nach dem vor oder nicht.

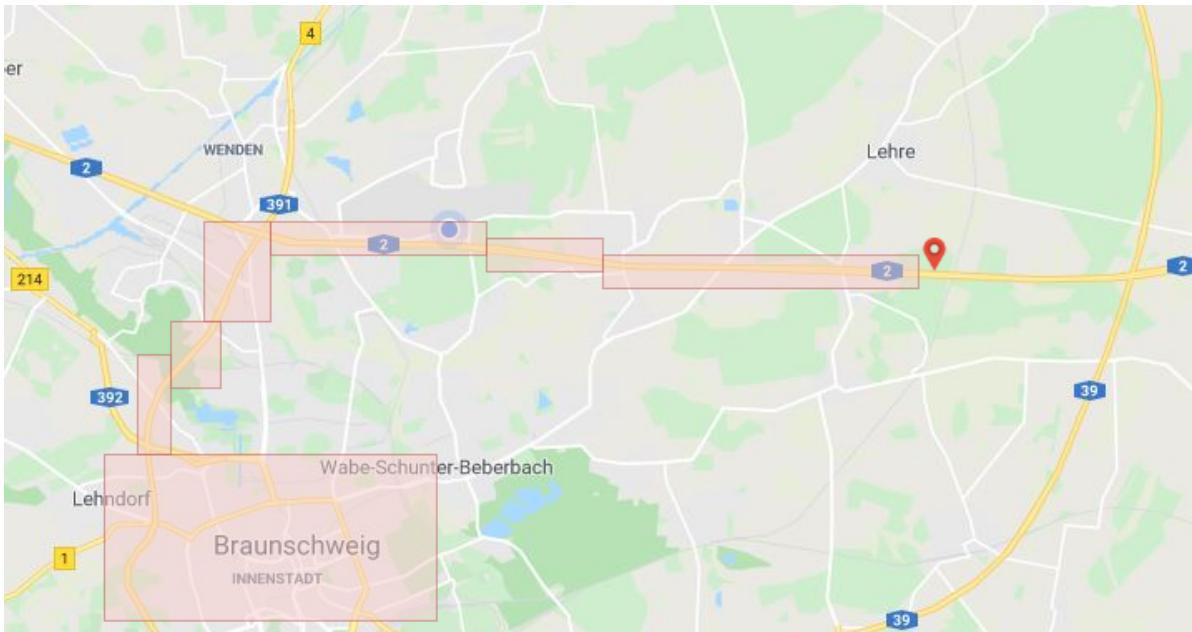


Abbildung 3: Umweltanalyse

B: Suche auf Basis der Fahrtenhistorie

Hierbei werden Muster in den zurückgelegten Strecken des Fahrers gesucht, um so dessen meist gefahrene Strecken zu identifizieren. Das Auto soll abschätzen können, wann der Fahrer welche Strecke zurücklegen wird. Vor dem jeweiligen Fahrtantritt soll die Suche nach Software identisch zu Variante A durchgeführt worden sein. Der Unterschied ist, dass die Suche hierbei selbstständig erfolgen soll, was ein besseres Nutzungserlebnis für den Fahrer schafft.

Ist die Suche (*egal ob gezielte Suche oder Umweltanalyse*) abgeschlossen, kann die Software dem Fahrer

zum Kauf vorgeschlagen und bei Bestätigung für das Auto bereitgestellt werden. Das nächste Kapitel verdeutlicht den Umfang der Bereitstellung.

2.3 Bereitstellung von Software

Wurde der Bedarf einer neuen Software vom Fahrzeug festgestellt, muss diese im folgenden bereitgestellt werden. Zuvor muss der Softwarezulieferer die Sicherheit von Softwarepaketen verifizieren und eine sichere Kommunikation zwischen Servern und Fahrzeugen herstellen. Der Server muss alle vorhandenen Softwarepakete verwalten und auf Anfrage das am besten geeignete Softwarepaket für das jeweilige Fahrzeug identifizieren und ein Angebot an dieses schicken. Das Fahrzeug muss die eingegangenen Softwareangebote über die Mensch-Maschine-Schnittstelle zu einem Zeitpunkt anbieten, an dem die Kaufbereitschaft des Fahrers möglichst hoch ist. Hierzu bedarf es einem Softwaremanagement, welches die Angebote des Servers verarbeitet. Da die Grundvoraussetzung der Wertschöpfungskette eine sichere Kommunikation zwischen Auto und Softwarelieferanten ist, wird zunächst eine mögliche Architektur zur sicheren Kommunikation vorgestellt.

2.3.1 OTA-Aktualisierungen mit UPTANE

Der im folgenden vorgestellte Standard "UPTANE", stellt eine Architektur und Vorgehensweise für die sichere Kommunikation zwischen einem Server und einem Auto vor. Hierdurch können Software-Aktualisierungen an intelligente Fahrzeuge sicher verteilt werden [?]. Erste Schritte hierzu wurden 2010 getätig, als Justin Samuel, Nick Mathewson, Roger Dingledine und Justin Cappos "*The Update Framework*" (*TUF*) entwickelten. Dieses bildet den Grundbaustein für den späteren "UPTANE"-Standard. Mittlerweile ist es Teil des "Automotive Grade Linux Projekts" und somit Teil der "Linux Foundation". Uptane stellt mittels einer Mehrschichtenarchitektur sicher, dass im Rahmen von Aktualisierungen keine Schädlingssoftware auf ein Auto gelangt. Zunächst wird die Architektur von Uptane dargestellt(Abb. 4).

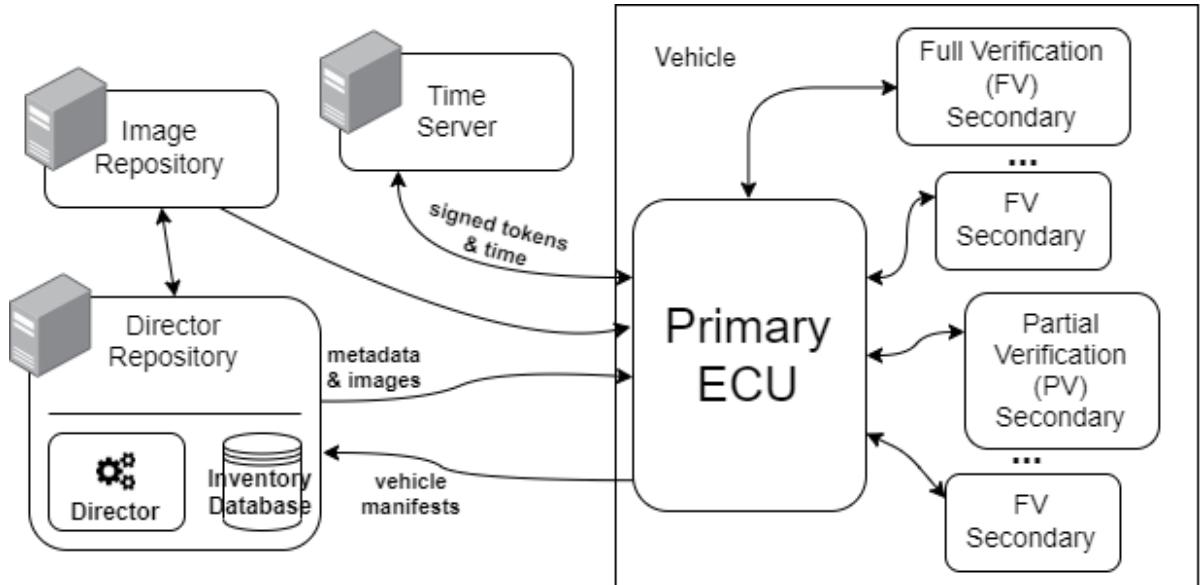


Abbildung 4: Architektur Design UPTANE [?]

Es ist vorab wichtig zu erwähnen, dass Uptane lediglich der Standard ist und keine offizielle Implementierung bereitstellt. Beispielhafte Implementierungen wären aktualizr⁴, rust-tuf⁵, Notary⁶ oder die OTA Community Edition⁷. Kommerziell entwickelte Systeme werden bereitgestellt von HERE Technologies⁸ sowie von Airbiquity⁹.

Die rechte Seite des Bildes stellt das Fahrzeug dar, die Elemente zur linken die Repositories. Diese Repositories sind Server und sie haben alle eine eigene wichtige Aufgabe. Der Time-Server ist dafür da, um ECUs über die aktuelle Zeit zu informieren, da viele ECUs keine Uhr haben [?]. Das Image Repository speichert jedes derzeit vom Lieferanten verteilte Image zusammen mit den Meta-Daten, welche zur Authentizität benötigt werden. Es nutzt Offline-Schlüssel um alle Metadaten zu "unterschreiben" bzw. zu verifizieren, was einen hohen Sicherheitsvorteil darstellt. Das Director Repository entscheidet abhängig von den übergebenen Informationen des Autos genau, welche Images an die ECUs verteilt werden müssen. Ein Auto hat mehrere ECUs, welche sich in Speicherplatzgröße, Stromverbrauch und Aufgabenbereich unterscheiden. Die *Primary ECU* verwaltet und steuert die Installationen auf den anderen ECUs.

In dem ersten Schritt einer Aktualisierung schickt das Fahrzeug sein Manifest an das Director Repository. Dieses enthält Informationen darüber, welche Images aktuell auf dem Auto installiert sind. Das Director Repository entscheidet anhand dessen, welche Software auf dem Fahrzeug installiert/aktualisiert werden soll. Die Metadaten und die neuen Images werden zurück an die ECU geschickt. Hier findet zunächst eine Verifikation der neuen Images statt, bevor sie anschließend bei erfolgreichem Test installiert werden. Die Verifikation der ECUs kann entweder vollständig oder teilweise erfolgen. **Vollständig** heißt in diesem Kontext, dass die Größe der Software und die Hashes, welche die ECU über die Metadaten vom Director Repository erhält, identisch sind zu denen der Metadaten die vom Image Repository zur Verfügung gestellt werden. Bei der **teilweisen** Verifikation muss lediglich die Signatur der Metadaten des Director Repositories mit der Signatur der Metadaten vom Image Repository übereinstimmen.

2.3.2 Anpassung von UPTANE Architektur

Damit die durch UPTANE bereitgestellte Architektur die Erkennung und Bereitstellung neuer Fahrumfänge unterstützt, muss diese dementsprechend angepasst werden, damit neben dem Aktualisieren bereits installierter Software auch das Installieren neuer Software möglich ist. Hierzu wird die Architektur von UPTANE erweitert und angepasst.

Damit das Director Repository nicht nur bestimmen kann, welche Software eine Aktualisierung benötigt, sondern auch welche auf dem Fahrzeug installiert werden muss um den Bedarf zu decken, braucht es eine geeignete Kommunikationsgrundlage. Hierzu soll das zuvor vorgestellte OpenSCENARIO XML-Speicherformat verwendet werden. Um die Suche nach neuer Software für das Directory Repository zu ermöglichen, wird das in Abbildung 5 dargestellte "*Scenario Repository*" eingeführt. In diesem werden OpenSCENARIO-Dateien gespeichert, welche als Basis für die Suche nach Software dienen. Jede Datei hat hält mindestens **eine**

⁴<https://github.com/advancedtelematic/aktualizr>

⁵<https://github.com/heartsucker/rust-tuf>

⁶<https://github.com/theupdateframework/notary>

⁷<https://github.com/advancedtelematic/ota-community-edition/>

⁸<https://www.here.com/products/automotive/ota-technology>

⁹<https://www.airbiquity.com/product-offerings/software-and-data-management>

Referenz in Form eines Schlüssels auf eine Software aus dem Image Repository.

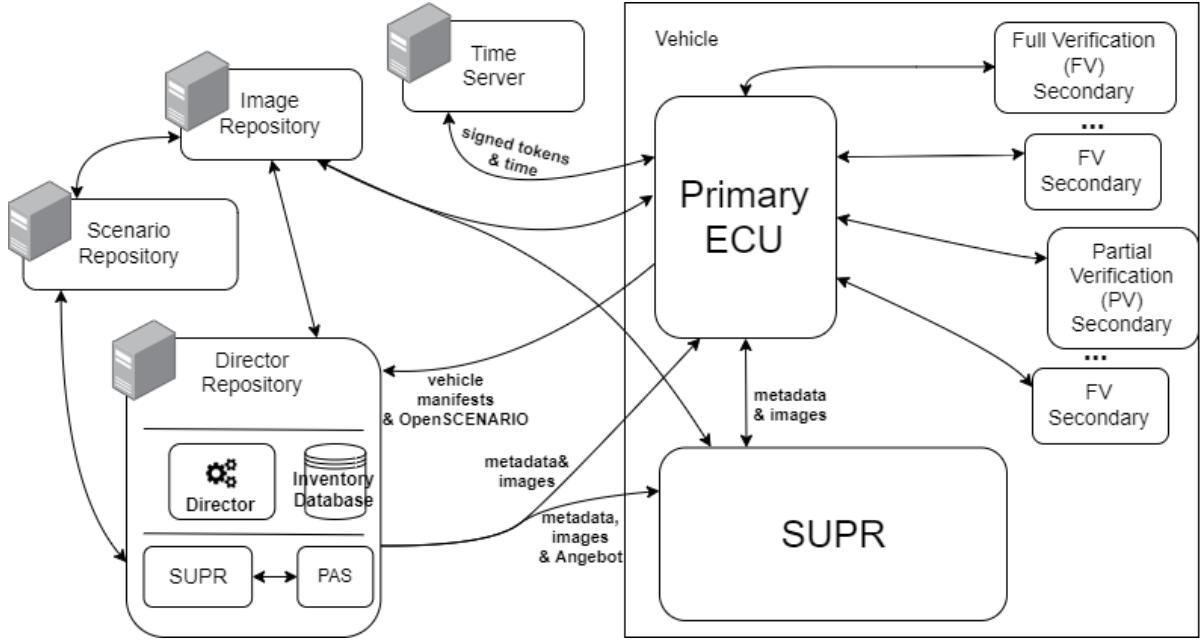


Abbildung 5: Angepasste Uptane Architektur

Dies bedeutet, dass die jeweilige Situation der OpenScenario-Datei von eben dieser Software bewältigbar ist. Erhält das Director Repository nun eine Anfrage für eine neue Software, wird diese anhand der mitgeschickten OpenSCENARIO-Dateien gesucht. Bei erfolgreicher Suche, soll diese in das Scenario Repository eingetragen werden. Die Suche führt der in Kapitel 2.3 eingeführte Software-User Pattern-Recognizer durch. In Kapitel 3.4 wird dieser um die Aspekte der Bereitstellung von Software erweitert.

Fahrzeughalter müssen neue Software kaufen, weshalb im Fall einer gefundenen Software zusätzlich zu den Metadaten und dem Softwarepaket (*Softwareimage*) noch eine Sammlung verkaufsbezogener Daten verschickt werden muss. Diese wird hier und im folgenden **Angebot** genannt. Ein Angebot muss alle möglichen Verkaufsarten (*Kauf, Miete, o.A.*) sowie dazu passende Preise beinhalten. Da die Preise und Verkaufsarten für Software je nach Automarke und -model variieren können sollen, wird ein Modul "pricing and sales", kurz PAS, zur Bestimmung des Preises und der Verkaufsart vorgeschlagen (siehe Abbildung 5). Das Director Repository ist nach wie vor dazu in der Lage, Softwareimages und Metadaten direkt an die Primäre Recheneinheit (ECU) zu schicken, wenn es sich nur um eine Aktualisierung bereits installierter Software handelt. Wird hingegen eine neue Software vorgeschlagen, so werden Image und Metadaten zusammen mit einem Angebot an den Software-User Pattern-Recognizer des Autos verschickt. Alternativ wird nur das Angebot verschickt um abzuwarten, ob der Kunde die Software tatsächlich haben will. Hierdurch würde das Image nicht unnötiger Weise verschickt werden, was unter anderem die Ressourcen schont. Der SUPR des Autos identifiziert unter stetiger Beobachtung des Fahrers, wann dieser einen Softwarevorschlag verarbeiten kann. Er stellt zum gegebenen Zeitpunkt Verbindung zu der Nutzeroberfläche des Wagens her, um so mit dem Fahrer zu kommunizieren und den Kaufvorgang abzuschließen (*Ob erfolgreich oder nicht ist hierbei egal*). Die Funktionsweisen und Aufgaben des SUPR im Auto unterscheiden sich von denen des SUPR im

Director Repository. Aufgrund dessen ist es wichtig eine Abgrenzung und Spezifizierung dieser vorzunehmen.

2.3.3 SUPR im Rahmen der Softwarebereitstellung

Das folgende komplettiert den in Kapitel 2.3 vorgestellten Software-User Pattern Recognizer (*SUPR*). Wie im ersten Teil wird auch im Rahmen der Bereitstellungskomponente des SUPR zwischen den Aufgaben auf Server und denen im Fahrzeug unterschieden.

SUPR im Director Repository

Die unterschiedlichen Suchvarianten welche der SUPR im Rahmen der Bedarfserkennung bereits auf dem Server ausführt, wurden in Kapitel 2.3 spezifiziert. Im Rahmen der Software Bereitstellung ist es die wesentliche Aufgabe, das Angebot und alle möglichen Preise festzulegen. Die verschiedenen Preise ergeben sich aus Kombination der einzelnen **Einflussfaktoren**.

Ein Angebot, welches vom Director Repository (siehe 3.2) an das SUPR des Fahrzeugs geschickt wird, spezifiziert die Preise für Software in Abhängigkeit der **Kaufart**, der **voraussichtlichen Laufzeit** des Mietverhältnisses (bei Miete), der Menge der Software welche der Fahrzeughalter in dem Moment akquirieren möchte und den eigentlichen Herstellungskosten der Software. Die folgende Tabelle definiert die Werte, welche die einzelnen Einflussfaktoren annehmen können.

Wert	Beschreibung
Kaufart	
Kauf	Der Fahrer akquiriert die Software dauerhaft. Eine Reklamation ist möglich, wenn diese ihre Aufgabe nicht korrekt ausführen würde.
Leihe	Der Fahrer legt einen Zeitraum(<i>von .. bis</i>) fest, für welchen er diese Software akquirieren möchte. Für diesen bezahlt er im Voraus und die SW wird bei Ablauf des Zeitraums deinstalliert.
Miete/Abo	Der Fahrer wählt die Dauer einer Mietphase, an welche sich der Preis anpasst. Das Mietverhältnis wird nach Ablauf dieses Zeitraums aufrecht erhalten. Das heißt es entstehen laufend Kosten bis es vom Fahrer beendet wird.

Zeitraum/Dauer	
Permanent	Der Zeitraum ist nur dauerhaft, wenn die Software gekauft wird. Hierbei tritt der höchste Preis auf.
Lang	Sechs Monate oder mehr. Auf einen Tag heruntergerechnet der günstigste Miet- oder Leihpreis.
Mittel	Sechs Wochen bis zu sechs Monate. Auf einen Tag heruntergerechnet ein wenig teurer als der des Langen.
Kurz	Acht Tage bis sechs Wochen. Auf einen Tag heruntergerechnet ein wenig teurer als der des Mittleren.
Einmalig	Ein bis sieben Tage. Auf einen Tag heruntergerechnet das teuerste Angebot.

Menge	
Einzeln	Es wird nur eine Software zu dem Zeitpunkt angeboten. Hat keinen Einfluss auf den Preis.
Mehrere	Software wird im Bundle gekauft, was sich positiv auf den Preis auswirkt. Ein Bundle muss vom SUPR im Fahrzeug erstellt werden.
Flottenkauf	Es wird Software für mehrere Fahrzeuge gekauft. Dies kann für Unternehmen Sinnvoll sein, die Dienstwagen anbieten oder auch Autovermietungen etc.

Weitere Einflüsse auf den Preis	
Herstellungskosten	Die Kosten die im Laufe der (Weiter-)Entwicklung und der Wartung dieser Software entstanden sind.
Beliebtheit	Ist eine Software beliebt bzw. gefragt, steigt der Preis dieser. Die Beliebtheit kann entweder global gemessen werden oder regionsabhängig. Der Preisanstieg sollte dabei nicht zu hoch sein.
Mächtigkeit	Je mehr eine Software "kann", desto teurer sollte diese sein.

Nachdem der SUPR des Director Repositorys eine den Bedarf deckende Software gefunden hat, muss er die Referenz auf die Datei an das PAS-Modul (*siehe Abbildung*) schicken. Das PAS-Modul soll anhand der Software, einiger Informationen zum Kaufverhalten der Fahrer des Fahrzeugs sowie dem generellen Bedarf der Software ein personalisiertes *Angebot* erstellen und anschließend an das Fahrzeug schicken.

SUPR im Fahrzeug

Neben dem Server-seitigen SUPR hat auch der SUPR des Autos wichtige Funktionen im Rahmen der Bereitstellung von Software. Ist das Angebot im Auto angekommen, identifiziert dieser einen geeigneten Verkaufszeitpunkt und stellt zu diesem das personalisierte Angebot über die Mensch-Maschine Schnittstelle (MMS) dar. Dies meint den Zeitpunkt, an dem der Verkauf oder die Leihgabe von Software möglichst wahrscheinlich ist. Hierzu ist eine Beobachtung des Fahrers nötig, um anhand von dessen **Eigenschaften** wie bspw. Mimik, Gestik, Sprache oder dem Fahrverhalten das Stress-, Freude, oder Angstlevel zu deuten, aber auch um Müdigkeit oder Ablenkung feststellen zu können. Vor allem Faktoren wie "Stress, Freude und Angst beeinflussen unsere Meinung und Entscheidung." Vgl. [?, S.44]. So wird eine Software eher nicht gekauft, wenn der Fahrer in eine stressige Verkehrssituation überblicken muss, zum Beispiel wenn es dicht benetzt ist und man zur Zeit in einer unbekannten Gegend Auto fährt. Hingegen beeinflusst Freude einen eher dazu, einen Kauf zu tätigen. Angst ist für den Anwendungsfall des Verteilens neuer Fahrfunktionen divers zu deuten. Wenn der Fahrer Angst vor der Strecke hat die vor ihm liegt, muss der SUPR dies anders bewerten als Angst die Aufgrund anderer Einflüsse entsteht. Diese Beobachtung entspricht der dritten Richtlinie Mensch-zentrierter autonomer Fahrzeuge nach Lex Fridman. [?, S. 3] Nach der fünften Richtlinie Fridmans, soll ein Auto von dem Moment dem ersten Einstiegs des Fahrers auf diesen personalisiert werden. [?, S. 5] So ist es Sinnvoll, würde der SUPR persönliche Eigenschaften in seine Entscheidungsfindung zum geeigneten Verkaufszeitpunkt einfließen lassen.

Ein System eines Fahrzeugs soll laut Fridman Daten-orientiert sein [?, S. 3]. Das heißt, dass ein Fahrzeug aus den aufgenommenen Daten lernen soll. Fusioniert man die bei der Fahrerbeobachtung aufgenommene Daten mit Daten über den Verkaufserfolg, kann der SUPR aus seinen Handlungen lernen und sich so optimieren. Hierdurch werden mögliche den Kauf negativ beeinflussende Faktoren identifiziert (zB. Stress) und

der Fahrer so besser kennengelernt. Die Kaufbereitschaft des Fahrers soll in Folge dessen möglichst oft korrekt eingeschätzt werden.

Ist der Zeitpunkt bzw. Zeitrahmen identifiziert, stellt der SUPR über die Mensch-Maschinen-Schnittstelle das personalisierte Angebot dar. Der dargestellte Inhalt wird maßgeblich beeinflusst von den Aspekten der Verkaufspräzesspsychologie. Um die wesentliche Aufgabe des SUPR (*Beobachtung & Kommunikation mit Fahrer, Vermarkten von Software*) sinnvoll zu modellieren, werden die Einflussfaktoren für den dargestellten Inhalt der MMS anhand der Prinzipien der Verkaufspolitik nach Markus Reinke [?] erläutert.

Um einen Kunden vom Erwerb eines Produkts zu überzeugen, muss diesem etwas angeboten werden wodurch ein offenes Bedürfnis befriedigt. Dazu muss herausgefunden werden, welches Produkt geeignet ist, damit es so anschließend angeboten werden kann. Wie der SUPR die Bedürfnisse von Fahrern identifiziert ist in Kapitel 2.3 nachzulesen. Was bei dem Anbieten einer Software beachtet werden muss, wird in diesem Kapitel abgehandelt.

Kunden sind unterteilbar in Plus-, Chancen- und Minus-Kunden. [?, S. 10ff.] Minus-Kunden werden ein Produkt von Beginn an nicht kaufen wollen, so gut der Erwerb auch gestaltet sein wird. Plus-Kunden werden ein Produkt voraussichtlich kaufen. Um die Chancen-Kunden von einem Erwerb zu überzeugen, werden im Absatz von Unternehmen diverse Prinzipien angewendet. Es kann also die Schlussfolgerung gezogen werden, dass so gut der SUPR seinen Fahrer auch kennt und wie gut Produkte den Bedarf auch abdecken, der Kunde dennoch nicht immer eine Software erwerben wird. Um die Anzahl an Käufern möglichst hoch zu halten, soll der persönliche Nutzen den die SW für den Kunden hat dargestellt werden. So kann die Situation simuliert und dargestellt werden, in welcher das Fahrzeug den Softwarebedarf festgestellt hat.

Das Differenzprinzip [?, S. 19fff.]

Nach dem Differenzprinzip sollen unterbewusste Reize gesetzt werden, die zum Erwerb leiten. Zum Beispiel wird mit der teuersten Variante des Produkts geworben, damit der Kunde bei näherem betrachten des Angebots die tieferen Preise, also ein "kleineres Übel", entdeckt und sich darüber erfreut. Dabei soll nicht die Frage **ob** der Kunde bei einem kaufen möchte, sondern **was** der Kunden von einem kaufen will.

Stellt der SUPR das Angebot dar, soll eine Mietfunktion im Fokus stehen, zudem deutlich gemacht werden. Die Option "Software Kaufen" ist kleiner darzustellen zusammen mit dem jeweiligen Preis. Um dem Kunden die Möglichkeit zu lassen die Software nicht zu erwerben, muss ein dezenter Knopf in der Nutzeroberfläche sein, welcher den Angebotsprozess beendet. Durch die Unauffälligkeit kann der Kunde zum Kauf gelenkt werden bzw. zu der Frage **was** man kaufen möchte, nicht **ob**. Um einen unterbewussten Reiz zu setzen, kann eine Karte dargestellt werden wie häufig die Software in der Umgebung installiert wurde. Dies sollte allerdings nur geschehen, wenn die Nachfrage tatsächlich auffällig hoch ist. Hiermit würde auch der Nachahmungseffekt abgedeckt werden (*weiter unten*).

Das Do-ut-des-Prinzip [?, S. 33fff.]

Nach diesem Prinzip, wird "gegeben, damit du gibst". Der Kunde wird durch beispielsweise Gratis-Proben angelockt um zum Erwerb bewegt zu werden. Will er dennoch nichts kaufen, kann um Weiterempfehlung gebeten werden. Dieser Bitte wird der Kunde vrs. nachkommen, da er im Vorfeld etwas "gratis" erhalten hat. Die "*Zwei Schritte vor, einer Zurück Taktik*" hierbei anzuwenden ist sinnvoll. Das heißt, dass dem

Kunden zuerst die teuersten Produkte gezeigt werden, um ihn anschließend auch hier mit niedrigen Preisen anderer Produkte beeindrucken und halten zu können. Es sollte ist hierbei auch nicht das Ziel das teuerste Produkt zu verteilen, sondern das eigentliche Ziel ist eines mit tieferem Preis.

Der SUPR kann dieses Prinzip anwenden, in dem er dem Kunden zu neu erworbener Software eine Test-Version einer anderen Software "schenkt", welche nach einem bestimmten Zeitraum (4 Wochen) abläuft. Diese Software sollte einen bestehenden Bedarf der Kunden abdecken.

Das Konsequenzprinzip

Konsequenz im Handeln wird in den meisten Kulturen als eine positive Charaktereigenschaft gewertet, strahlt Sicherheit aus. Zudem schafft es Vertrauen bei Kunden. Konsequenz kann unter anderem mittels wenn-dann-Fragen ausgestrahlt werden, also "Wenn wir Ihnen XY bieten, kaufen Sie dann?" Der Kunde würde bei Erfüllung seines Bedarfs eher kaufen. Und da der von ihm benannte Bedarf abgedeckt ist, wird er so von einem frühen Commitment überzeugt. Es sollen die Wünsche und Prioritäten der Kunden erkannt werden und anschließend das Produkt mit Fokus auf diese beworben werden.

Dieses Prinzip sollte vor allem verwendet werden, um einen "Fuß in die Tür zu bekommen" und nicht um das teure Produkt zu verkaufen. Sinnvoll ist es, nach dem Motto "Kleinvieh macht auch Mist" zu verkaufen. Der SUPR tut dies zum einen indem er die Situation, in welcher der Bedarf festgestellt wurde, auf der MMS darstellt und den Kunden so erinnert. Wenn der Kunde noch gar keine oder nur wenig Software erworben hat, soll ihm immer die Möglichkeit einer kostenlose Probe-Woche geboten werden. Dies soll bis zu fünfmal geschehen, danach wird ohne weiteres keine Gratis Software mehr angeboten.

Das Nutzen des Nachahmungseffekts [?, S. 67fff.]

"Der Mensch ist grundsätzlich ein Gemeinschaftswesen und achtet daher sehr darauf, was andere von ihm denken, sowohl in der breiten Öffentlichkeit als auch in seinem engen persönlichen Umfeld." - Markus Reinke (Vgl.) [?, S. 67]

Dieses Verhalten, der sogenannte **Nachahmungseffekt**, kann durch das Einsetzen verschiedener Techniken provoziert werden. Bei der **Zeugenumlastung** lässt sich der Kunde bestätigen, dass ihr Unternehmen und Produkt "gut" ist indem er von anderen positives darüber in Erfahrung bringt. Eine Variante, wie diese vom SUPR angewendet werden kann, wurde zuvor bereits dargestellt. Bei der **Referenztechnik** werden die Stammkunden eines Unternehmens gebeten Empfehlungsschreiben zu erstellen mit welchen im folgenden geworben werden kann. Der SUPR kann hierzu ein Bewertungssystem einführen und Nutzer können die Bewertung anderer Fahrer lesen und sich so eine Meinung der Software einholen. Dabei ist es sinnvoll, einige vorgefertigte Antwort/Bewertungsmöglichkeiten selber zur Verfügung zu stellen. Eine dritte Technik ist das **Empfehlungsmarketing**, bei welchem Kunden gebeten werden uns an Freunde und Verwandte weiter zu empfehlen. In den zeiten von Social Media bietet es sich an, eine "Teilen"-Funktion zur Verfügung zu stellen. Der Fahrer soll Teilen können, wie viele Kilometer er in welcher Zeit zurückgelegt hat, wie viele davon autonom bewältigt wurden und welche Softwarepakete genutzt wurden.

Um eine hohe Resonanz herzustellen, ist es sinnvoll dem Kunden ähnlich zu sein, sympathisch zu wirken, Lob & Anerkennung zu verteilen als auch andere zu unterstützen. Da es ungewohnt ist, Lob oder anderes von einem Computer zu erhalten, der SUPR aber dennoch eine hohe Resonanz erzielen soll, sollte den Kunden

eine Bezugsstelle gegeben werden. Eine Referenz aus der Wirtschaft hierzu stellt "Meet Olli" dar [?]. Olli ist das Maskottchen des UserInterfaces (UI) von "Local Motors" (LM). LM stellt einen autonom fahrenden Bus her, mit welchem die Insassen über eine UI interagieren können. Olli hat "nur" zwei Augen und einen Mund, stellt hierdurch aber einen vermenschenlichten Bezugspunkt für den Fahrer und die Insassen dar und gewinnt so das Vertrauen der Insassen.

Darüber hinaus soll der Kunde nicht von den Angeboten "erschlagen" werden, sondern der SUPR muss diese dosiert vorlegen. Wenn der Kunde kein Interesse hat, ist dies zu respektieren und er soll damit nicht weiter konfrontiert werden. Der SUPR soll nur die tatsächlich benötigte Software bewerben - ein einmalig aufgetretener Bedarf fällt da nicht drunter (*Die Ausnahme ist, der Kunde ist ein absoluter Plus-Kunde*). Es muss genügend Werbung betrieben werden damit möglichst viel Software verkauft wird. "Klassische" Werbung auf der MMS ist nicht Sinnvoll, sondern eher Werbung in Form von Gratis-Proben von Software-Paketen. Nutzt der Fahrer eine dieser Proben und die SW kommt zum Einsatz, kann der SUPR den Fahrer über die MMS darüber in Kenntnis setzen, wodurch er vor Augen geführt bekommt, dass er diese Software braucht.

Ein Auto wird allerdings nicht nur von einer Person gefahren, sondern möglicherweise von mehreren Familienmitgliedern, Mitarbeitern oder anderen Mietern im Falle von Carsharing. Das SUPR muss dazu in der Lage sein, diese einzelnen Personen auseinanderhalten zu können um keine falschen Entscheidungen zu treffen.

Wird dem SUPR ein Softwarevorschlag mit Angebot zugeschickt, soll dieser bis das Angebot angezeigt werden kann, diese Software schon anfangen herunterzuladen. Dadurch kann die Software, wenn sie erworben wird, bereits während der Fahrt installiert werden, was den Verkauf schneller und Nutzerfreundlicher gestaltet.

2.3.4 Die Mensch-Maschine-Schnittstelle

Die Mensch-Maschine-Schnittstelle ist das System, über welches der Fahrer und das Auto mit einander interagieren. Im Kontext intelligenter Fahrzeuge handelt es sich hierbei meist um einen Bildschirm, mitunter haben Fahrzeuge auch einen integrierten Sprachassistenten.

Das Bereitstellen neuer Fahrfunktionen schließt mit ein, dass ein Fahrzeug zum Zeitpunkt A selbstständig Fahren kann und zum Zeitpunkt B nicht. Zwischen diesen Zeitpunkten muss eine Übergabe der Fahraufgabe von dem Fahrzeug an den Fahrer erfolgen. Dabei soll der Fahrer des Fahrzeugs über die momentane Verkehrslage ausreichend in Kenntnis gesetzt werden, um während und nach der Übergabe der Fahraufgabe die Sicherheit zu wahren. Der Begriff ausreichend definiert sich dabei wie folgt:

Die Inkenntnissetzung gilt als vollständig, wenn die Mensch-Maschine-Schnittstelle alle Prinzipien von S. Debernard et Al. [?] erfüllt.

Das zu entwickelnde Display soll Nutzer-zentriert (User-Centered) entwickelt werden, um die Anforderungen möglichst vieler Nutzer erfüllen können und zusätzlich den Faktor der "Vermenschlichung" zu erhöhen. Die geschaffene transparente Straßenführung über das Display eines Fahrzeugs, soll zusätzlich durch Audio-Assistenten unterstützt werden. Die MMS wird so durch zu einem audiovisuellem System.

Die Schnittstelle soll sich sowohl visuell als auch auditiv an die im Fahrzeug sitzenden Personen anpassen. So

kann zum Beispiel die Schriftgröße größer sein, wenn eine Person im Rentenalter das Fahrzeug betritt oder ähnliches. Mögliche Features der Personalisierung (*z.B. größere Icons, dunkles Design*) werden mit Hilfe der Personas 2.4.1 in der Bachelorarbeit erarbeitet. Im generellen soll der Nutzer auf dem Display eine Simulation des eigenen Autos sehen zusammen mit zusätzlicher Information gemäß den Prinzipien nach Debernard et. Al. Das Design des User-Interfaces kann vom Fahrer angepasst werden, wobei die intuitive Bedienung sich nicht verschlechtern soll. Durch die Vermenschlichung des MMS, einer Transparenten Straßenführung, der Möglichkeit die Steuerung jederzeit zu übernehmen sowie der höflichen Kommunikation mit dem Fahrer, wird das Vertrauen des Fahrers zu dem Fahrzeug gefördert [?], was den Kauf von Software wahrscheinlicher macht.

2.4 Technologie- und Methodik-Scouting

Um auf Grundlage der dargestellten Rahmungen für die Bedarfserkennung und Bereitstellung neuer Fahrumfänge für intelligente Fahrzeuge einen Prototypen zu entwickeln, wird abschließend ein Technologie- und ein Methodik-Scouting durchgeführt werden. Das Kapitel der Arbeitsmethodik spezifiziert Den Zyklus des User-Centered-Designs und stellt dar, weshalb dieser für die Entwicklung des Prototypen die richtige ist. Das anschließende Technologie-Scouting stellt Entwicklungsumgebungen, Frameworks und Programmiersprachen vor, welche in der Entwicklung genutzt werden.

2.4.1 Arbeitsmethodik: User-Centered-Design

Um die Nutzeroberfläche für die Zielgruppen ansprechend zu gestalten, wird bei der Erarbeitung dieser nach den Prinzipien des User-Centered Designs (*UCD*) gehandelt. Ins deutsche übersetzt bedeutet dies "Benutzerzentriertes Design bzw. Benutzerorientierte Gestaltung" eines Produkts. Es beschreibt einen Designprozess bzw. ein Entwicklungsverfahren, "auf welches der Endnutzer (Fahrer) schon von Anfang an Einfluss nimmt" (Vgl. [?, S. 763]). Der User wird in die Phasen der Entstehung einer Benutzungsschnittstelle integriert was zur Folge hat, "dass der Aufbau, die Inhalte und deren Form sowie das Design des Endproduktes maßgeblich von den Bedürfnissen, Erwartungen und dem Verständnis der User bestimmt wird" (Vgl. [?]). Des weiteren kann durch das Einbeziehen von Kunden die Qualität optimiert werden (Vgl. [?, S. 14]).

"Typisch für einen UCD-Prozess zum Entwerfen von Webanwendungen mit optimaler User Experience sind [jedoch] die Prozessschritte Analyse, Konzeption, Umsetzung/ Design, Evaluierung und Optimierung." (Vgl. [?])

Im Rahmen der Analyse werden die Anforderungen an das System erstellt und zusammengeführt. Hierzu können Personas erstellt werden (siehe: 2.4.1) und aus eben diesen können Anforderungen erstellt werden. Die Analyse soll zudem "Usability" [?] Ziele festlegen, anhand welcher in der "Evaluierung und Optimierung" die Güte der Nutzeroberfläche gemessen werden kann [?]. Während der Konzeption soll das Verständnis der Benutzer und deren Bedürfnisse hinsichtlich der User Experience auf die Benutzeroberfläche übertragen werden [?, (ebd.)]. Die Konzeption endet mit dem erstellen eines Prototypen. Dieser wird in der Dritten Phase (Design) durch "ein konsistentes, ansprechendes und klares Grafik Design"(Vgl.) [?] erweitert, was Probleme löst und eine Intuitive Bedienung unterstützt. Die Phase der "Evaluierung und Optimierung" wird das Produkt auf Probleme wie Sicherheit oder eine schlechte Bedienbarkeit hin untersucht. Hierdurch werden

Mängel entdeckt welche in den nächsten Zyklus der Entwicklung einfließen.

Wie hoch der Grad der Usability und der User-Experience ist, lässt sich wie zuvor erwähnt anhand von Personas ableiten. Im Folgenden wird daher erklärt, was Personas sind und es werden eigene Personas für das Projekt erstellt.

Personas

Bei Personas handelt es sich nun um fiktive Personen, also hypothetische User, mit individuellen Eigenschaften [?,], welche ebenso eine reale Person haben könnte. Je ähnlicher die Personas also der Zielgruppen des Unternehmens/des Produktes sind, desto effektiver ist die Verwendung dieser.

Personas sind nicht nur als potentielle Zielgruppe zu sehen - sie haben zudem die Aufgabe, dass sich das Entwicklerteam in die Position des Nutzers versetzen kann. Dazu werden Personas zunächst nach Kategorien wie "Geschlecht" oder "Alter" aufgeteilt und anschließend wird jedes Profil mit Merkmalen und Eigenschaften gefüllt. Welche Informationen unter anderem zu Personas hinzugefügt werden, zeigt die folgende Tabelle.

Vor- und Nachname	Geburtsdatum/Alter
Foto der Person/Aussehen	Herkunft/Wohnort
Sprachkenntnisse	Beruf; Berufserfahrung in Jahren
Bildung/Ausbildung	Familienstand
Interessen und Hobbys	Fähigkeiten/Behinderungen
Sicherheitsrisikofaktoren für den Straßenverkehr	Abneigungen
Erfahrungen mit Technik (Meidenkompetenz)	Vorlieben
Fahrerfahrung	Kaufbereitschaft
Auto; Wenn ja: Besitzverhältnis?	

Anhand dieser möglichen Informationen werden im folgenden die Personas für dieses Projekt erstellt.

Name / Soziographische Daten	Hobbys & Interessen	Anwendungsfallbezogenes
 <ul style="list-style-type: none"> • Dietmar Müller, 68 Jahre (Bild: [?]) • Loppersum, Ostfriesland • Deutsch • Gelernter Maurer • Verheiratet, 3 Erwachsene Kinder 	<ul style="list-style-type: none"> • Rentner; Taxiunternehmer(seit 20 Jahren aktiv) • Fischen, Wandern • Handwerklich begabt, Hobbygärtner • Diabetiker • idR. Minus-Kunde • Geizig 	<ul style="list-style-type: none"> • Führerschein seit 47 Jahren • Vertraut Technik nicht • Schlechte Sehkraft • Ängstlicher Autofahrer • Besitzt eigenen Neuwagen
 <ul style="list-style-type: none"> • Jake Schneiders, 52 Jahre (Bild: [?]) • Washington, USA • Englisch, Russisch(Fließend) • Ausgebildeter Cyberhacker • Geschieden, ein Kind (geteiltes Sorgerecht) 	<ul style="list-style-type: none"> • Leutnant beim Militär • Jagen, Baseball Trainer • Kindersorgerecht unter der Woche • Workaholic • Chancen-Kunde 	<ul style="list-style-type: none"> • Führerschein seit 27 Jahren • Vertraut sehr auf Technik • guter, sicherer Autofahrer • Setzt Wert auf gute Bedienbarkeit • Hat gerne die Kontrolle über Systeme • Fährt einen Wagen des Militärs • Arbeitet viel am Laptop

 <ul style="list-style-type: none"> • Chi Nguyen, 24 Jahre (Bild: [?]) • Rom, Italien • Vietnamesisch, Italienisch(Fließend), Englisch (Fließend) • Studentin (Geschichte) • Single 	<ul style="list-style-type: none"> • Studentin, Stadtführerin in Rom • Joggen, Roadtrips • Influencer • Plus-Kundin 	<ul style="list-style-type: none"> • Führerschein seit 2 Jahren • Vertraut sehr auf Technik • unsichere Autofahrerin • Ist mit viel Technik aufgewachsen • Mag es, ihre Apps zu personalisieren • Nimmt Teil an Car-Sharing
---	---	---

Die erstellten Personas werden bei der Erstellung des Prototypen herangezogen um Anforderungen an das System. Jede einzelne ermöglicht es, bestimmte Schwierigkeiten und Herausforderungen genauer darzustellen. So soll Dietmar Müller verdeutlichen, dass ein Kunde möglicherweise doch vom Kauf einer SW überzeugt werden kann, obwohl er idR. ein Minus-Kunde ist. Durch ihn kann auch dargestellt werden, welche weiteren Einflussfaktoren es neben "Stress" o.Ä. gibt, wie bei seine Diabetes Erkrankung.

Jake Schneiders soll die Art Person darstellen, welche neugierig ist, die Sicherheit eines System testen möchte und hierzu auch selber in der Lage ist. Durch seine technische Affinität kann er Systeme leicht Verstehen und neue Anforderungen erstellen, die voraussichtlich leicht zu implementieren sind.

Chi Nguyen ist eine sehr feminine Studentin welche bereits vielerorts gewohnt hat. Durch ihr Dasein als Influencerin hat sie ein großes Netzwerk an Followern, mit welchen sie ihre Erfahrungen die sie während der Autofahrt sammelt Teilen kann.

Da alle Personas in unterschiedlichen Ländern an dem Straßenverkehr teilnehmen, sind die Einflussfaktoren zur Erstellung von Anforderungen sehr vielfältig, was sich positiv auf die Entwicklung auswirken kann.

2.4.2 Tech-Scouting

Damit die Entwicklung des Prototypen möglichst simpel sein wird, wird im Folgenden ein Technologie-Scouting durchgeführt. Es soll die Vorteile aufzeigen, welche ausgewählte Entwicklungsumgebungen, Programmiersprachen und Frameworks für die Entwicklung des Prototypen haben.

Im Rahmen eines Prototypen für die Erkennung und Bereitstellung neuer Fahrfunktionen für intelligente Fahrzeuge ist es nötig, ein auf der Straße fahrendes Fahrzeug zu Simulieren und dazu eine passende Nutzeroberfläche zu haben, welche als Mensch-Maschine Schnittstelle des Fahrzeugs agiert. Für das Simulieren einer Straßenverkehrssituation wird der OpenSource Simulator **Carla**¹⁰ verwendet. Mit Carla ist es möglich, das eigene Auto zusammen mit anderen Fahrzeugen, Radfahrern, Fußgängern, Hindernissen uvm. auf der Straße darzustellen. Autos können mit Sensoren ausgestattet werden und die aufgenommenen Daten können mittels der Python-API ausgelesen werden. Der Ablauf einer Simulation wird in einem Python-Script spezifiziert. Eine alternative Möglichkeit zur Erstellung einer Simulation ist das einbinden einer OpenSCENARIO-Datei. Es existiert ein Aufsatz auf Carla, wodurch der Simulator den in einer solchen Datei dargestellten Ablauf selber darstellen kann.

Für die Erstellung von Python Skripts wird "**PyCharm**"¹¹ als Entwicklungsumgebung gewählt. PyCharm ermöglicht eine intuitive und einfache Anbindung an GIT. Da die aus Carla ausgelesenen Daten verarbeitet werden müssen um zu erkennen wann das Auto nicht mehr selbstständig fahren kann, ist die Entwicklung eines Servers notwendig. Neben der Analyse der Simulation soll dieser als Kommunikationszentrale zwischen eben diesem und der Mensch-Maschine-Schnittstelle dienen. Zur Entwicklung des Servers soll das ebenfalls von JetBrains entwickelte **IntelliJ IDEA**¹² dienen. Auch IntelliJ bietet eine einfache Anbindung an GIT an - zusätzlich ist die Einbindbarkeit von **Maven**¹³ zum verwalten von *Dependencies* ein leichtes.

Für die Entwicklung der Mensch-Maschine-Schnittstelle wird eine Android App entwickelt - für diese wird **Android Studio**¹⁴ verwendet. Android Studio ermöglicht neben der visuellen auch eine textuelle Bearbeitung von Nutzeroberflächen. Es ist die von Google empfohlene Entwicklungsumgebung und ermöglicht das erstellen von Apps massiv.

Neben Python wird zum entwickeln von Server und MMS hauptsächlich die Programmiersprache Java (Version 1.10) verwendet. Java wurde an der Universität Oldenburg gelehrt und es wird unter anderem zum entwickeln von Android Apps verwendet.

2.5 Ausblick und erstes Konzept der praktischen Umsetzung

Nachdem in den vorherigen Kapiteln einzelne Aspekte der Wertschöpfungskette erläutert wurden, soll es abschließend einen Ausblick auf die Bachelorarbeit geben und ein erstes technisches Konzept des Prototypen vorgestellt werden. Im Laufe der Arbeit haben sich einige zu erläuternde Aspekte aufgetan, welche aus Platzgründen erst in der Bachelorarbeit behandelt werden.

Im Rahmen dieser soll zunächst ein erster Zyklus des UCD durchgeführt werden, um so die Nutzeroberfläche der Mensch-Maschinen-Schnittstelle zu erarbeiten. Dabei sollen zusätzlich mögliche Personalisierungsfeature benannt werden. Weiterhin sollen die anderen im Folgenden vorgestellten Teilsysteme entwickelt werden. Unter anderem soll ein Suchalgorithmus für den SUPR des Servers erarbeitet werden sowie eine

¹⁰<http://carla.org/>

¹¹<https://www.jetbrains.com/de-de/pycharm/>

¹²<https://www.jetbrains.com/de-de/idea/>

¹³<https://maven.apache.org/>

¹⁴<https://developer.android.com/studio>

Gegenüberstellung von Rechenleistung und Sendeleistung des Autos bzw. des Servers erfolgen. Es werden die einzelnen Bausteine einer Wertschöpfungskette (zur Erkennung und Bereitstellung neuer Fahrumfänge intelligenter Fahrzeug) benannt, erläutert und im Prototypen veranschaulicht. Im Folgenden wird für diesen ein erstes technisches Konzept vorgestellt.

Erstes technisches Konzept des Prototypen

Abbildung 6 zeigt eine erste Architektur des Prototypen. Dieser ist unterteilt in das **Fahrzeug**(unten) und den **Server**(oben). Das Fahrzeug ist in drei Teilsysteme aufgeteilt. Der "Carla Python Skript Client" ist das Modul in dem die Simulation stattfindet. Entscheidungen bzgl. der Fahraufgabe werden in diesem getroffen. Die **Mensch-Maschine-Schnittstelle** ist eine Android-basierte App. Sie wird über einen Emulator bedienbar sein und soll unter anderem Informationen aus dem Carla Simulator darstellen. Damit dies möglich ist, wird der dritte Baustein des Fahrzeugs eingeführt: der SUPR.

Der SUPR des Fahrzeugs soll die im Forschungsseminar definierten Aufgaben durchführen können und dient zusätzlich als Kommunikationseinheit. Er verbindet die Carla Simulation und die Mensch-Maschine-Schnittstelle, sodass eine interne Kommunikation im Fahrzeug möglich ist. Desweiteren dient der SUPR als Kommunikationseinheit zum Server. Er ist für die Installation eingehender Softwarevorschläge zuständig und muss entsprechende Updates an den Python Client senden. In der Uptane Architektur waren hier die ECU's eines Autos angedacht. Da diese in dem ersten Konzept nicht auftauchen, steuert dies der SUPR.

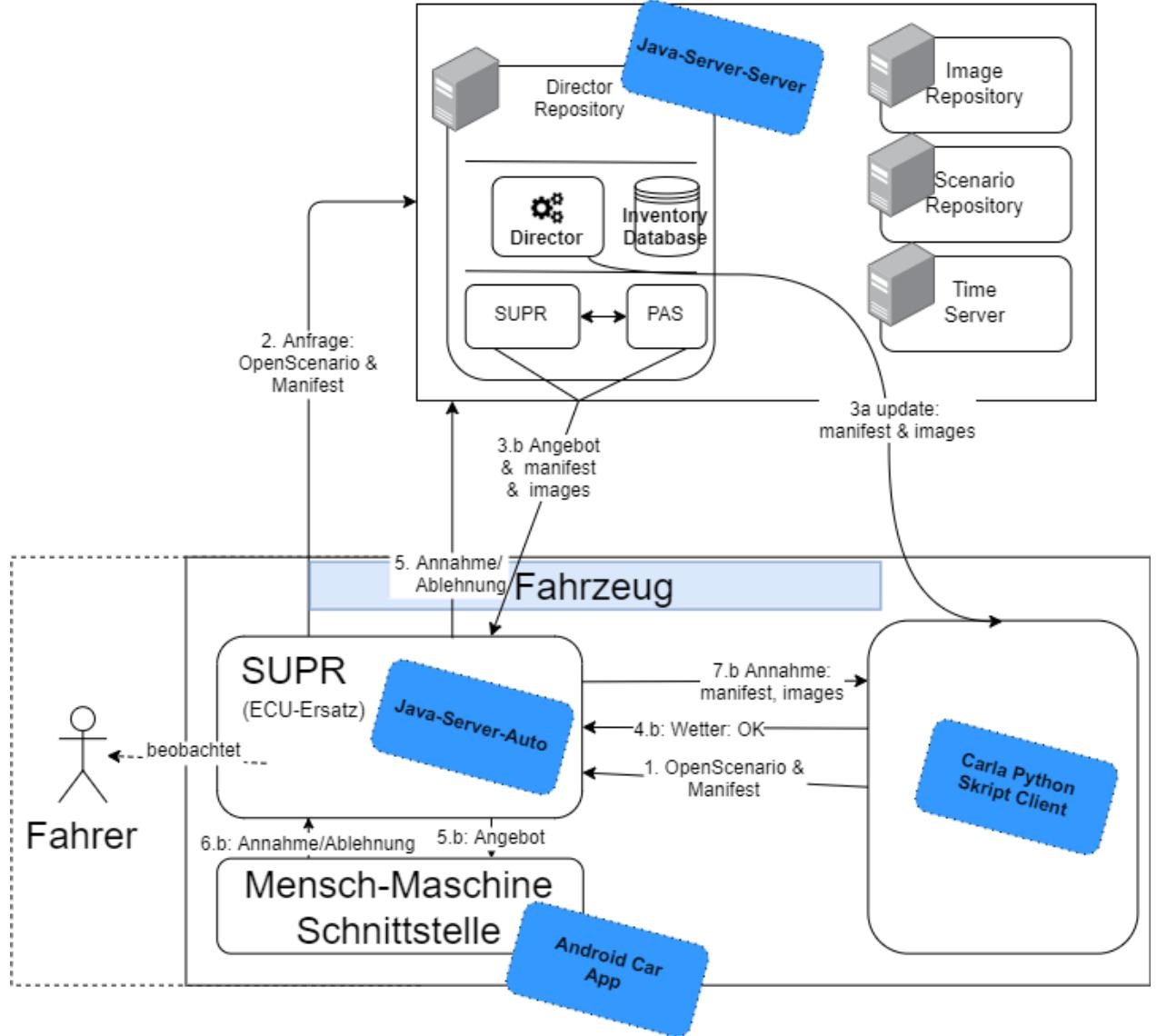


Abbildung 6: Technisches Erstkonzept

Die Architektur des Servers ist identisch zu der aus Kapitel 3.2. In der Implementierung sollen der Einfachheit halber die vier Bestandteile des Servers auf einem Server laufen und nicht getrennt werden. Der SUPR des Director Repositorys soll die im Forschungsseminar ihm zugeschriebenen Aufgaben durchführen können, gleiches gilt für den PAS. Die einzelnen Bestandteile des Servers können mit Ausnahme des Time-Servers miteinander kommunizieren.

Aus den Pfeilen des Konzepts abzuleiten ist der mögliche Ablauf eines Kaufprozesses. Initiiert wird dieser dadurch, dass das Fahrzeug im Carla Client nicht mehr autonom Fahren kann. Dieser erstellt infolgedessen eine OpenScenario-Datei und schickt diese gemeinsam mit dem Manifest, in welchem die bereits installierte Software notiert ist, an den SUPR (Pfeil 1). Dieser schickt den entdeckten Bedarf als Anfrage an den Server

(Pfeil 2), welcher infolgedessen überprüft welche Software den entdeckten Bedarf abdecken kann. Hierzu sucht der Director des Servers im Manifest des Fahrzeugs nach zu aktualisierender Software. Wird ein Defizit festgestellt, stellt der Server eine Verbindung mit dem Python Client her, welcher dieses Update installiert (Pfeil 3.a). Ist jede Software wieder aktuell stellt der SUPR des Servers fest, ob die Bedarfssituation mit den Updates bewältigt werden kann. Ist dies **nicht** der Fall, sucht der SUPR nach der passenden Software anhand der OpenScenario Datei. Ist ein Paket gefunden, wird das vom PAS erstellt Angebot zusammen mit dem unterschriebenen Manifest und dem Software-Image an den SUPR zurückgeschickt (Pfeil 3b).

Wird eine neue Software vorgeschlagen, hält der SUPR den Vorschlag so lange von der Nutzeroberfläche fern, wie der Fahrer sich noch in einer 'stressigen' Situation befindet. Ob dies der Fall ist, wird anhand des Wetters abgeleitet. Ist es regnerisch, nebelig oder ähnliches, soll das Angebot zurückgehalten werden. Bei sonnigem Wetter darf es dann auf der Nutzeroberfläche angezeigt werden. (Pfeil 4.b) Der Fahrer wird anhand einer Mitteilung darüber informiert, dass eine neue mögliche Software vorhanden ist. Der Fahrer kann das Angebot in der Nutzeroberfläche anpassen und es abschließend entweder annehmen oder ablehnen(Pfeil 6.b). Wird das Angebot angenommen, installiert der SUPR (Java-Client) die Software auf dem Fahrzeug, indem er eine Nachricht an das Python Skript schickt. Die Simulation wird erneut gestartet und das Auto kann die zuvor schwierige Situation bewältigen.

3 Das Business Model Canvas und Wertschöpfungskette

Damit die Verteilung von **Software** geeignet organisiert wird, bedarf es einer Plattform über welche Softwares orts- und zeitunabhängig heruntergeladen werden können. Diese Plattform kann in Form eines Softwareshops realisiert werden, über welchen Softwares von Fahrzeughaltern gekauft und installiert werden können. Der Shop stellt den Mittelsmann zwischen Fahrzeughaltern und den Softwareherstellern dar. Die installierten Softwares können unter anderem den Fahrumfang autonomer Fahrfunktionen erweitern oder das Auto mit anderen Akteuren des Straßenverkehrs verbinden. Ein bereitgestellter Shop sollte Automarkenübergreifenden entwickelt werden, da hierdurch zugleich ein großer Teil des Marktes gewonnen wird und außerdem viele Mehrwerte für Fahrzeughalter entstehen, wie die folgenden:

1. Nachhaltigkeit von Fahrzeugen

Nach dem verlassen des Fließbandes altert die Software eines Autos.¹⁵ Updates sind heutzutage nur beim Mechaniker möglich und ist zudem mit einem großen Aufwand verbunden. Durch eine Kabellose Schnittstelle sollen Fahrzeughalter gewünschte Software orts- und zeitunabhängig installieren können. Durch Softwares können Fahrzeuge sicherer (*weniger Unfälle*) und schonender (*geringere Getriebebeanspruchung etc.*) gefahren werden¹⁶, wodurch sich die Lebenszeit des Fahrzeugs verlängern könnte. Bleiben Fahrzeuge länger "*modern*", kann die Nachfrage an Neuwagen langfristig zurückgehen kann. Ist der Shop Automarken-übergreifend, können sämtliche Software des Shops für jeden Kunden weltweit verfügbar sein. Hierzu müssen sämtliche Fahrzeuge die Softwares installieren und kompilieren können.

2. Selbstständige Erweiterung des Fahrzeugs

Die schlechte Alternative zu einer kabellosen Bereitstellung von Software ist die stetige Fahrt zum Mechaniker um neue Softwares zu installieren. Hier erhält ein Fahrzeug eine festgelegte Sammlung an Softwares, der Fahrzeughalter hat also keine Auswahlmöglichkeiten. Hierdurch kann auf Dauer viel Software auf dem Fahrzeug installiert sein, die der Fahrzeughalter nicht benötigt. Durch einen Softwareshop wird es möglich, dass ein Fahrzeughalter nur die tatsächlich benötigte Software auf seinem Fahrzeug installiert kann. Des weiteren können die Kosten für Fahrzeughalter hierdurch skallierbar gehalten werden, da ein Fahrzeughalter selber entscheiden kann welche Softwares er kauft.

Um einen Überblick des Marktes zu geben, wird zunächst ein Business Model Canvas (*BMS*) erarbeitet. Anschließend werden relevante Bausteine der Wertschöpfungskette anhand der Erkenntnisse aus dem Forschungsseminar sowie des Business Models identifiziert und deren Aufgaben erläutert. Abschließend erfolgt eine Zusammenfassung des Kapitels und es werden die technischen Konzepte vorgestellt.

3.1 Business Model Canvas

Das 2004 von Alexander Osterwalder entwickelte Busines Model Canvas (*BMC*) schafft einen Überblick über die Aufgaben, die Kosten- und Partnerstrukturen sowie den Kundensegmenten eines Unternehmens. Es hilft den Fokus auf die wesentlichen Zielsetzungen dessen zu setzen.¹⁷ Folgend werden die Kundensegmente,

¹⁵quelle

¹⁶quelle

¹⁷<https://ut11.net/de/blog/dein-geschaeftsmodell-kompakt-der-business-model-canvas/>

die Kundenbeziehungen, die Marketingkanäle und die Einnahmequellen betrachtet. Anhand dieser werden anschließend die Nutzenversprechen sowie die Schlüsselressourcen und -aktivitäten bestimmt. Durch die abschließende Bestimmung von Schlüsselpartnern und der möglichen Kostenstruktur eines Softwareshops wurden "alle wesentlichen Elemente eines Geschäftsmodells in ein skalierbares System gebracht."¹⁸

3.1.1 Kundensegmente

Die Kundensegmente des Marktes lassen sich in Einzelkunden, Gruppen und Flottenbetreiber aufteilen. Flottenbetreiber lassen sich in zwei Segmente aufteilen: "*Leihe und Leasing*" umfasst Autovermieter, Unternehmen die ihren Mitarbeitern Leasingwagen bereitstellen, aber auch weitere wie Car-Sharing Unternehmen. Deren Kunden erwarten eine grundlegende Sammlung an Software im Mietfahrzeug vorzufinden und wollen selbstständig weitere Softwares auf eigene Rechnung installieren können. Neben Leihe und Leasing sind auch Unternehmen mit Firmenwagen ein gesondertes Kundensegment. Beide haben kleine oder große Fahrzeugflotten und müssen Softwarekäufe dementsprechend skalieren können.

Einzelkunden sind die übliche Autofahrer, die ein privates Fahrzeug besitzen und Software auf diesem installieren möchten. Gruppen sind mehrere Einzelkunden, die gemeinsam Software kaufen um hierdurch Kosten zu sparen. Hierdurch werden Familienkäufe aber auch Käufe mit Freunden möglich. Durch ihre enorme Größe ist es sinnvoll, auch dieses Segment weiter zu unterteilen.

- 18-25** Die Generation Z und deren ältere Geschwister sind mit modernen Technologien wie dem Computer und dem Smartphone aufgewachsen. Für sie ist es intuitiv zu Alltagsunterstützenden Hilfestellungen zu greifen. Zugleich stellt dies jedoch die Gruppe mit dem geringsten Einkommen dar. Deshalb sollte sie nicht nur zum Kauf von Software bewegt werden, sondern auch für die Vermarktung in den älteren Kundensegmenten genutzt werden.
- 25-33** Ebenfalls bestens mit Technik vertraut, umfasst dieses Segment die vermutlich wichtigsten Kunden. Die Präsenz und Vernetzung im Internet dieser Gruppe ist ähnlich zu dem vorherigen Segment sehr hoch. Personen dieser Gruppe sind in der Regel dabei, sich ein geordnetes Leben aufzubauen in Form einer Karriere, einer Familie oder anderem. Sie sind finanziell besser aufgestellt als die jüngeren Segmente, können also quantitativ mehr Software beziehen. Sicherheit, Datenschutz und Privatsphäre spielen für dieses Segment eine wichtige Rolle, da sie die schlechten Seiten einer Vernetzten Welt kennen und hierdurch vorsichtig agieren. Durch ihr fortgeschrittenes Alter sind sie in den Augen älterer Generation Vertrauenswürdig und möglicherweise der Ansprechpartner im Bezug auf Technik.
- 33-50** Fest im Leben stehend, stellt dieses Segment das Mengenmäßig größte dar. Im Gegenteil zu den jüngeren Segmenten ist hier wahrscheinlich dass die meisten ein eigenes Auto haben (*was in den vorherigen Gruppen nicht immer der Fall ist*). Das Auto ist hier ein Statussymbol und ein Alltagsgegenstand für die Vielzahl an Kurzstrecken die zurückgelegt werden.
- 50-65** Auch in diesem Segment haben die meisten ein eigenes Auto. Dieses wird sich öfters geteilt, da die Notwendigkeit für Zwei Autos nicht mehr gegeben ist oder die mittlerweile erwachsenen Kinder das Auto der Eltern nutzen. Die Anforderungen sind vergleichbar zu denen der 33-50 Jährigen, jedoch lässt sich diese Gruppe im Bezug auf neue Technik eher beraten.

¹⁸<https://www.startplatz.de/startup-wiki/business-model-canvas/>

65-75 je älter der Kunde ist, desto geringer ist die durchschnittliche Technikaffinität. In diesem Segment sind die Mehrwerte von Technik maßgebend dafür, ob ein Kauf stattfindet oder nicht. Eröffnet sich Raum zur Kritik, neigen diese eher vom Kauf ab. Zeitgleich sind sie von den jüngeren Segmenten einfach zu beeinflussen wenn es um den Kauf neuer Technik geht.

75 + Durch das fortgeschrittene Alter stellt dieses Segment oftmals eine große Gefahr für den Straßenverkehr dar. Es legt Wert auf ein weitgehend selbstständig fahrendes Fahrzeug, da so Strecken zurückgelegt werden können die im Normalfall zu unsicher zurückzulegen wären.

3.1.2 Kundenbeziehungen

Über die Wichtigkeit des Einzelkundensegments ist eine einheitliche Behandlung von Kundensegmenten wichtig. Damit Fahrzeughalter nach einem Erstkauf auch weitere Transaktionen in Betracht ziehen, müssen deutlich erkennbare Mehrwerte geliefert werden. Unterstützend hierfür sind die Grundbausteine der Kundenbeziehung *Vertrauen aufzubauen, Unterstützung bei Kaufentscheidung bereitstellen, Individuelle und zuvorkommender Kundenservice*. Die Akquise von Software soll *fließend* sein, um nicht als Akt hohen Aufwands wahrgenommen zu werden.

Der Aufbau einer guten Kundenbeziehungen kann gut durch gratis Software oder Software-Gutscheine initialisiert werden. Die ersten Akquisitionen sollten einen deutlichen Mehrwert für den Käufer bieten, um eine möglichst hohe Zufriedenheit bei diesem zu erzielen. Damit eine positive Kundenbeziehung bestehen bleibt, wird die Verwaltung und Überwachung von Software teilweise durch die Systeme übernommen. Werden Softwares nicht genutzt oder es ergeben sich Möglichkeiten für den Fahrzeughalter Geld zu sparen, weißt der Softwareshop hierauf hin. Eine langfristige Kundenbindung kann durch die Einbindung dessen in das Brainstorming für neue Softwares zu integrieren. Kunden können so ihre Wünsche erfüllt bekommen und es werden ihre größten Wünsche im Bezug auf ihr eigenes Fahrzeug können in Erfüllung gehen.

Da Flottenbetreiber mit einem Softwarekauf größere Summen ausgeben und somit auch ein größeres Finanzielles Risiko eingehen, bedarf es einen höheren Grad der individuelle, nicht automatischem, Beratung. Hierdurch können Fehler präventiert werden und somit die Zufriedenheit des Segments sichern.

3.1.3 Marketingkanäle

Um weitblickend die marktführende Plattform für die Erkennung und Bereitstellung von Fahrfunktionssoftware zu sein, sollte ein Markteintritt auf allen möglichen Marketingkanälen stattfinden. Die Altersgruppen des Einzelkundensegments werden über unterschiedliche Kanäle erreicht. Um ältere Segmente (50+) zu erreichen, ist das Nutzen *traditioneller* Medien wie Plakate, Zeitschrift oder dem Fernsehen Sinnvoll. Zu Sendezeiten, in denen überwiegend junge Zuschauer aktiv sind, sollen die Werbespots dementsprechend abgeändert werden. Jüngere Kunden sind aktiver auf anderen Medien oder Plattformen, wie Instagramm, Facebook, Twitter, aber auch und vor allem Youtube. Über diese kann zum einen Influencermarketing (*Influencer bewerben die Plattform*) betrieben werden oder auch klassische Werbeanzeigen geschaltet werden. Im Fokus stehen Menschen, die permanent unter Zeitdruck sind, da sie durch ein autonom Fahrendes Fahrzeug viel zeit einsparen können.

3.1.4 Nutzenversprechen

1. Orts- und Zeitunabhängige Akquise eigens ausgewählter Fahrfunktionen

Statt wie in der Vergangenheit Softwareupdates für das Fahrzeug nur beim Mechaniker installieren zu können, können Fahrzeughalter über den Softwareshop immer und überall Software akquirieren. Die installierten Funktionen können selbstständig gewählt werden, wodurch Fahrzeuge individuell gehalten werden können.

2. Überwachung, Verwaltung und Vorschlagen von Software

Fahrzeughalter sollen überwachen können, welche Software wie oft genutzt wird nicht benötigte Softwares einfach deinstallieren zu können. Neben dieser Unterstützung bei der Entdeckung nicht Benutzer Software, wird auch die Suche nach einem möglichen Softwarebedarf für Fahrzeughalter automatisiert. Dies kann vor allem weniger technikaffine Kundensegmente unterstützen und Sicherheit im Umgang mit dem Fahrzeug schaffen. Wie der Bedarf einer Software erkannt werden kann, wird in Kapitel 2.2.3 erläutert.

3. Autofahren wird sicherer

Software fährt sichere als Mensch, da keine äußeren Einflüsse wie Stress, Müdigkeit, Wut oder anderes die Fahrsicherheit negativ beeinflusst. Neben den Sicherheitsvorteilen einzelner Fahrsysteme eines Fahrzeugs ist weitblickend vor allem die Sicherheit, welche durch C2X-Kommunikation ermöglicht wird der entscheidende Faktor der die Sicherheit weiter erhöhen wird.

4. Lebenszeit des Autos wird verlängert

Aufgrund der gestiegenen Sicherheit wird ein Fahrzeug künftig unfallfreier Fahren. Des Weiteren wird die Fahraufgabe von einer Software schonender für Getriebe- und Motorteile durchgeführt. Beides ermöglicht eine durchschnittlich längere Lebenszeit für ein Fahrzeug, was auch die Anschaffung eines neuen Fahrzeugs verzögern kann.

5. Geringerer Wertverlust von Fahrzeugen

Durch das stetige Nachladen von Fahrfunktionen kann ein Fahrzeug auf dem aktuellen Stand der Technik bleiben. Hierdurch werden Unterschiede bezüglich der Fahrfunktionen von Fahrzeugen überschaubar und der Wertverlust eines Fahrzeugs ist dementsprechend geringer.

6. Fahrzeughalter gewinnen Zeit

Durch das Nutzen autonomer Fahrfunktionen können immer weitere Strecken zurückgelegt werden ohne die Steuerung zu übernehmen. Die hierdurch gewonnene Zeit kann zum Arbeiten, oder auch zum verbringen von Zeit mit den anderen Insassen des Fahrzeugs verwendet werden. Diese Zeit war vorher durch die Fahraufgabe belegt.

7. Stetige Erweiterung des Softwareangebots

Durch die Bereitstellung eines offenen Marktes für Softwareanbieter, wird der entstehende Wettbewerb die Entwicklung neuer Softwares mit unterschiedlichen Funktionen antreiben. Das Spektrum an möglichen Softwares wird zunehmend größer werden und mehr "Probleme" des Alltags bewältigen.

8. Interaktion mit der Autoumwelt

Bestimmte Softwares ermöglichen die Kommunikation mit anderen Akteuren der Umwelt (*Ampeln, Parkplätze, andere Fahrzeuge, Drive-In-Restaurants, uvm.*) und schaffen so neue Möglichkeiten miteinander zur Interaktion mit diesen über das Auto.

3.1.5 Einnahmequellen

Der Softwareshop hat Zwei unterschiedliche Einkommensströme. Zum einen kann es eine 'Anmeldegebühr' geben, die Softwareprovider zahlen müssen um Software im Shop veröffentlichen zu können. Außerdem können diese ihre Softwares im Shop bewerben, indem sie Werbeflächen kaufen auf denen ihre Software zum Kauf angeboten wird. Dies ist Vergleichbar mit dem Ergebnis einer google-Suche, in welcher Ganz oben *gesponserte* Internetseiten/Produkte zu finden sind.

Neben Einnahmen durch die Softwareprovider wird auch Umsatz durch den Kauf und die Nutzung von Software generiert. Beim Kauf einer Software geht ein fester Prozentsatz des Verkaufspreises an den Betreiber des Shops. Zum Vergleich: Bei Android Apps liegt der Anteil den Google einnimmt bei 30%. Auch bei entstehende In-App-Käufen (*z.b. für die Nutzung eines Services*) geht ein Anteil von 10% an den Shop. Die vorgestellten Kundensegmente interagieren dementsprechend nur mit Software- und Serviceprovidern.

Durch die Integration von Entwicklungs- und Analyse-Tools können weitere Einnahmen generiert werden. Provider können diese nutzen und werden durch sie bei der Entwicklung neuer Softwares unterstützt.

3.1.6 Schlüsselressourcen

Die wichtigsten Ressourcen sind die **geschaffene Plattform des Softwareshops** und die dort vertriebenen **Softwares**, welche von Fahrzeughaltern gekauft werden können. Vor allem ein starkes und effizientes Backend ist wichtig, um große Mengen an Softwares und sonstiger Daten zu speichern. Damit der Faktor der Ortsunabhängigkeit von Softwaredownloads realisiert werden kann, ist ein stabiles Kommunikationsnetzwerk (z.B. 5G) notwendig. Softwares werden zum Großteil von Software Providern bereitgestellt. Ohne die stetige Erweiterung des Angebots, wird der Shop weniger gut laufen als mit.

Eine weitere Schlüsselresource ist die Einbindung eines elektronischen Zahlungsservices, damit Fahrzeughalter tatsächlich kaufen können.

3.1.7 Schlüsselaktivitäten

Die Schlüsselaktivitäten sind die Aufgaben des Unternehmens, die zur Erfüllung vorgestellter Nutzenversprechen führen. Sie sind an die Schlüsselressourcen und Nutzenversprechen gekoppelt und sollen die Anforderungen dieser erfüllen. Im Kontext des Softwareshops können die Aktivitäten in Vier Gruppen unterteilt werden.

1. Aktivitäten der Bedarfserkennung von Software

- Software Klassifizierung
- Automatische Erkennung möglichen Softwarebedarfs

2. Aktivitäten der Bereitstellung von Software

- Sicherheitsverifikation von Software anhand von Sicherheitskonzepten
- Sicherer Download über das Internet bereitstellen
- Bereitstellung einer elektronischen Zahlungsschnittstelle
- Angebotsunterbreitung
- Eigenständige Softwareentwicklung

3. Aktivitäten zur Steigerung der Kundenzufriedenheit

- Shop Verwaltung
- Shop (*Weiter-)Entwicklung und Wartung*
- Eigenständige Verwaltung von Software
- Eigenständige Überwachung von Software
- Kundenservice und individuelle Beratung

4. Aktivitäten zur Verbesserung der Supply Chain

- IDE Entwicklung
- Dokumentation & API Reference
- Tool Entwicklung (*Für Entwickler*)

3.1.8 Schlüsselpartner

Es gibt verschiedene Schlüsselpartner:

- **Softwareprovider**

Zur schnellen Bereitstellung von Software auf der Plattform müssen in kurzer Zeit möglichst viele Softwares hinzugefügt werden. Softwareprovider müssen verstehen, wie eine Software für das Fahrzeug aufgebaut sein muss und bei dem Lernprozess unterstützt werden. Eine zusätzlich bereitgestellte IDE für die Entwicklung von Softwares (*Vergleich hierzu: Android Studio¹⁹*) kann die Provider bei diesem Lernprozess unterstützen.

- **Automobilhersteller**

Um einen großen Anteil des Marktes ist es sinnvoll auf möglichst vielen Fahrzeugmarken verfügbar zu sein. Hierzu sind Kooperationen mit Automobilherstellern notwendig, die die Plattform in ihre Fahrzeuge einbauen. Ohne derartige Übereinkommen, ist es nicht möglich die Plattform zu betreiben.

- **Mobilfunkbetreiber**

Wie zuvor erwähnt, ist ein gut ausgebautes Mobilfunknetz wichtig für die ortsunabhängige Bereitstellung von Software.

¹⁹ Android Studio quelle

3.1.9 Kostenstruktur

Durch das Entwickeln, Betreiben und Vermarkten des Shops entstehen vorwiegend Kosten im Rahmen der Bezahlung von Entwicklern und Managern. Zum Wachstum der Plattform werden immer mehr Entwickler benötigt werden, weshalb auch weiterführend die Bezahlung dieser eine großer Kostenpunkt ist. Weiter fallen Stromkosten für das Betreiben von Servern.

3.2 Konzept einer Wertschöpfungskette

Die in Abbildung 7 dargestellte Supply Chain zeigt den Weg von der Entwicklung zum Absatz von Software an Kunden. Software Provider stellen Software für den Shop bereit und verdienen an einer verkauften Software. Sie können mit Service Providern zusammenarbeiten um so neue Funktionalitäten in Fahrzeugen zu schaffen und so zusätzlichen Umsatz generieren. Die im Software Store bereitgestellten Softwares können über das Internet heruntergeladen und im folgenden von Kunden genutzt und verwaltet werden. Da

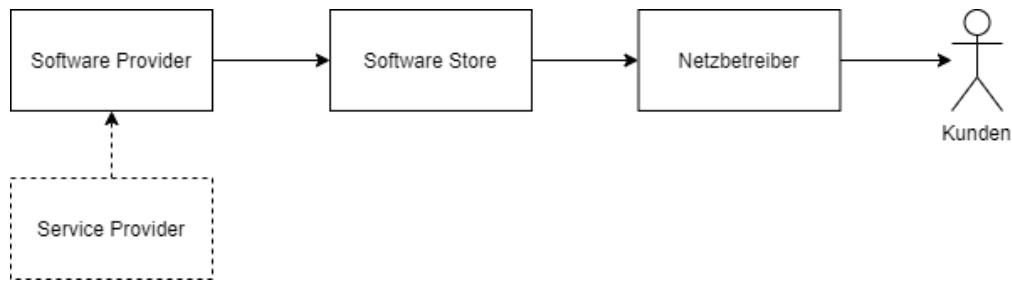


Abbildung 7: Supply Chain von Software

die Erstellung eines Software Stores viele Aufgaben mit sich bringt, wird im folgenden ein Konzept einer Wertschöpfungskette für diesen erstellt. Diese soll die einen Überblick über die Aufgaben de Stores bieten. Nach der Wertschöpfungskette von Porter sind Unternehmen in primäre und unterstützende Aktivitäten aufzuteilen.²⁰ Primäre Aktivitäten "liefern dabei einen direkten wertschöpfenden Beitrag zur Erstellung eines Produktes."²¹ Unterstützende Aktivitäten sind als "notwendige Voraussetzung zu Erstellung der Produkte"²² zu sehen, also Dinge die im Rahmen aller Primäraktivitäten wichtig sind. Abbildung 8 gibt einen Überblick der identifizierten Bausteine der einzelnen Aktivitäten, welche im folgenden genauer dargestellt werden. Im skizzierten Businessmodel wurde im Rahmen der Schlüsselaktivitäten bereits eine Unterteilung in unterschiedliche Aufgabenbereiche vorgenommen (*Kap. 3.1.7*), welche im folgenden für die Gruppierung der Bausteine verwendet wird.

3.2.1 Primäre Aktivitäten

Es gibt Fünf unterschiedliche Primäre Aktivitäten. Das Kaufen und Lagern von Materialien wird der **Eingangslogistik** zugeordnet. Im Kontext des Softwareshops gibt es hier Zwei wichtige Bausteine:

Software-Sicherheit verifizieren

Sämtliche Softwares, die in dem Shop aufgeführt werden sollen, müssen durch ein Sicherheitskonzept verifiziert werden. Es sollten mehrere Sicherheitskonzepte und Richtlinien erarbeitet werden, auch zwecks einer Unterscheidung von Fahrfunktions- und Service Software. Diese Konzepte unterstützen die Bereitstellung von (sicherer) Software.

Es muss sichergestellt werden, dass sämtliche erhobenen Daten entsprechend der Datenschutzrichtlinien des jeweiligen Landes verarbeitet werden. Neue Software muss außerdem auf Viren überprüft werden.

Im Fall von *Fahrfunktionssoftware* muss zudem sichergestellt werden, dass diese keine Sicherheitslücken

²⁰quelle

²¹<https://refa.de/service/refa-lexikon/wertschoepfungskette>, 21. Mai 2020

²²ebd.

in der Fahraufgabe aufweisen. Um dies zu erreichen, kann einer solchen Software eine Sammlung an OpenScenario Dateien beigefügt werden, die beschreiben welche Art von Situation durch die Software abgedeckt wird. Anhand dieser Dateien kann zum einen, wie in Kapitel 2.2.2 beschrieben, die Suche von Softwares durchgeführt werden. Zum anderen können diese als Simulationsgrundlage dienen. Simulationen können Aussage darüber treffen, eine Software in der abzudeckenden Situation Sicher handelt oder nicht. Sie sollte auch im Rahmen der automatischen Erkennung von Softwarebedarf genutzt werden.

Klassifizierung von Software

Erfüllt eine Software die Sicherheitsrichtlinen, wird sie anschließend Klassifiziert. Eine Klassifizierung soll die entsprechenden Metadaten zu Softwares zuordnen, welche für die Einordnung im Shop notwendig sind. Dieser Schritt hat einen Direkten Bezug zur Bedarfserkennung von Software. Anhand der Klassifizierung sollen Fahrzeughalter intuitiv wissen, wie gut/schlecht eine Software ist und welche Aufgabe sie erfüllt. Hierzu muss ein Konzept erarbeitet werden, welches die Bewertungsfaktoren festlegt. In Kapitel 4.1 wird eine Beispielkonzept vorgestellt.

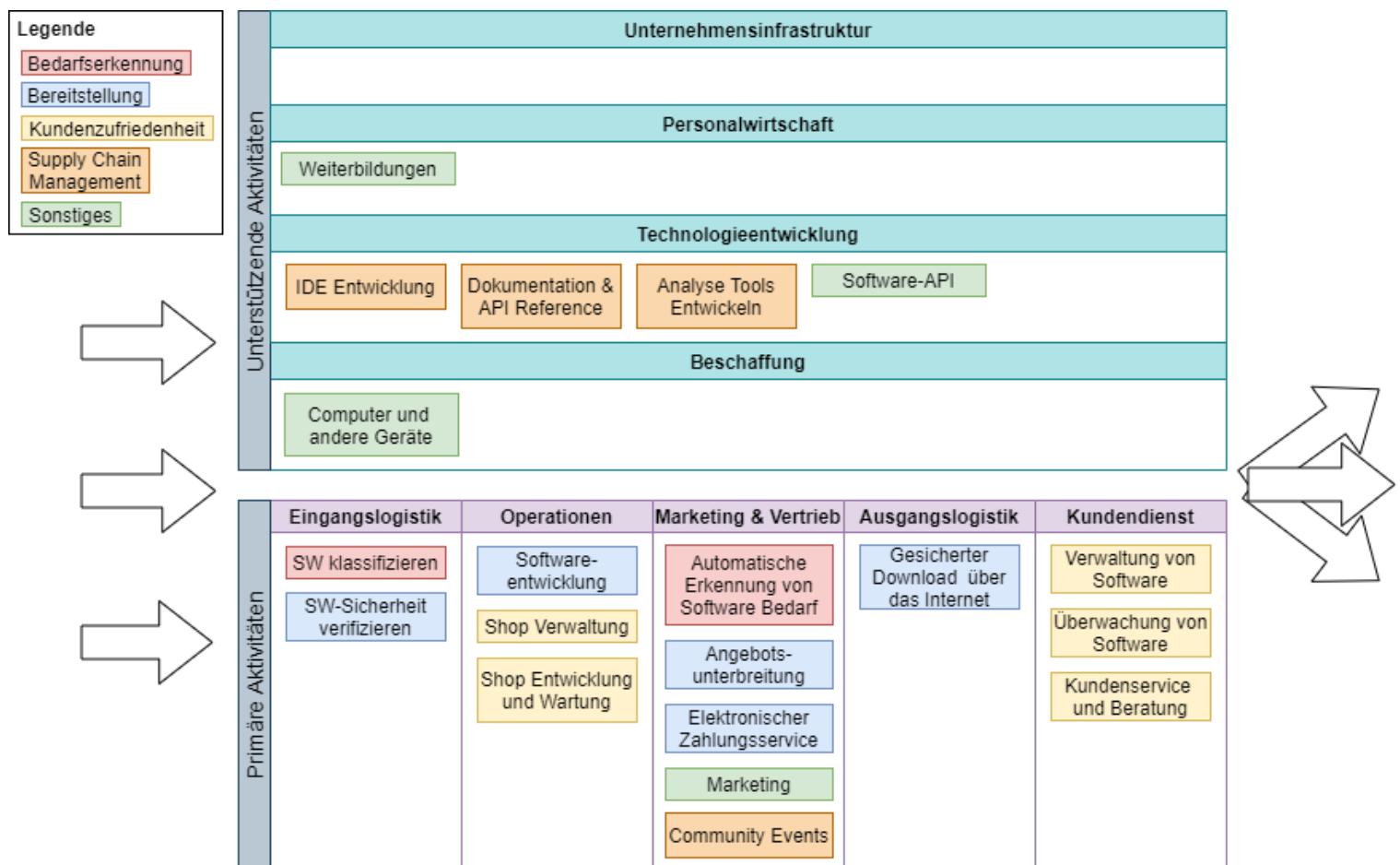


Abbildung 8: Die wichtigsten Bausteine einzelner Aktivitäten der Wertschöpfungskette

Nach erfolgreichen Durchlaufen der Aktivitäten der Eingangslogistik folgen die Aktivitäten der **Operation**.

Im Rahmen dieser werden die verifizierten und klassifizierten Softwares für den Absatz an Fahrzeughalter vorbereitet und auch absatz-unterstützende Aufgaben ausgeführt.

Eigene Softwareentwicklung

Neben der Bereitstellung des Shops kann zusätzlicher Umsatz durch den Verkauf eigener Software generiert werden. Hierdurch vergrößert sich nicht nur das Spektrum vorhandener Softwares, sondern es kann das allgemeine Image des Unternehmens positiv beeinflussen. Durch die innerbetriebliche Entwicklung kann auch der Verifikationsprozess der Software beschleunigt werden, was die Bereitstellung neuer Softwares ebenfalls früher ermöglicht.

Shop Verwaltung

Um die Kundenzufriedenheit weiter zu steigern, muss der Inhalt des Shops verwaltet werden. Einige Softwares sollen zeitweilig hervorgehoben werden, und auch die angezeigten 'vorgeschlagenen Softwares' müssen kontinuierlich an die Fahrzeugdaten angepasst werden. Softwares müssen hinzugefügt, entfernt oder aktualisiert werden und *Empfehlungen der Redaktion* können gesetzt werden. Anstößige Kommentare oder gefährliche Softwares müssen aus dem Shop entfernt werden.

Des Weiteren muss gesteuert werden, welche Softwares den Fahrzeughaltern vorgeschlagen werden. Dies kann durch die Einbeziehung von Fahrdaten der Fahrzeugs geschehen, da so Softwares entlang der häufig zurückgelegten Strecken präsentiert werden können,

Shop Entwicklung und Wartung

Auch die (Weiter-)Entwicklung und Wartung des Shops trägt zu einer höheren Kundenzufriedenheit bei. Änderungen im Frontend können die User Experience (UX) steigern und somit zum Kauf animieren. Bezuglich der Frontend Entwicklung sollten die Prinzipien des User Centered Designs beachtet werden. Änderungen des Backends können den Umfang des Shops erhöhen oder bisherige Funktionen (*bspw. das Vorschlagen von Software*) verbessern und kundenzentrierter gestalten.

Für jegliche Änderungen sollte außerdem stets auf die Community gehört werden, welche sich aus den Software Providern und den Kundensegmenten zusammensetzt. Vorschläge und Wünsche für neue Softwares seitens der Kunden sollen evaluiert werden. Diese Vorschläge können möglicherweise veröffentlicht werden und von Providern zur Ideenfindung genutzt werden. Die Zusammenarbeit mit Software Providern ist im Generellen sehr wichtig, weshalb im Rahmen der Unterstützenden Aktivitäten (3.2.2) einige Bausteine zur Förderung dieser vorgestellt werden.

Der Dritte Schritt einer Wertschöpfungskette ist zusammengesetzt aus dem **Vermarkten des Software Shops und dem Vertrieb von Software**. Im Rahmen dieser finden sich die Bereitstellung unterstützende Bausteine in Form von Angebotsunterbreitung und der Bereitstellung einer elektronischen Zahlungsschnittstelle wieder. Außerdem findet hier die automatische, fahrzeug-individuelle Erkennung von Softwarebedarf statt.

Automatische Erkennung von Software Bedarf

Um die Nutzenversprechen der *automatischen Erkennung eines Software Bedarfs* zu erfüllen, müssen Serverseitige Algorithmen diese anhand von Fahrzeug und Fahrzeughalterdaten identifizieren. Hierfür ist eine Implementierung des Uptane-Standards (*Kapitel 2.3.1*) notwendig.

Der Prozess automatischer Erkennung eines Software Bedarfs wird im folgenden skizziert. Zunächst muss hierfür das **Manifest des Fahrzeugs zusammen mit Fahrdaten an den Server geschickt werden**. Im Manifest befindet sich eine Liste aller installierten Softwares des Fahrzeugs. Die verschickten Fahrdaten treffen Aussage darüber, welche Strecken vom Fahrzeug oft zurückgelegt werden. Auf dem Server angekommen, wird **entlang dieser Strecken nach möglichen Softwares für das Fahrzeug** gesucht. Wurden Softwares gefunden, die noch nicht auf dem Fahrzeug installiert sind werden mehrere **virtuelle Instanzen des Autos** erstellt. Eine dieser Instanzen umfasst nur die Softwares, welche laut dem Manifest auf dem Fahrzeug installiert sind. Den anderen Instanzen wird jeweils eine Software hinzugefügt, die entlang der Strecken gefunden wurden.

Nach dem erstellen dieser virtuellen Fahrzeuge werden im vierten Schritt die jeweiligen Situationen der neuen Softwares zugeordneten OpenScenario-Dateien **simuliert** und von den **virtuellen Fahrzeugen durchlaufen**. Zum Vergleich durchfährt das virtuelle Fahrzeug welches keine neuen Softwares erhalten hat die gleichen Situationen durch. Anhand eines Vergleichs der Simulationen kann festgestellt werden, ob die Software positive, negative oder überhaupt keine Auswirkungen auf das Fahrverhalten des Fahrzeugs hat. Positiv wäre, wenn das Fahrzeug die Situation nun selbstständig durchfahren kann oder dies Ressourcenschonender tut. Negativ wäre, wenn die Situation durch die neue Software nicht weiter durchfahren wird oder die neue Software zum durchfahren der Situation nicht vom Fahrzeug ausgewählt wird. Dies kann bedeuten, dass auf dem Fahrzeug bereits eine Software installiert ist, welche diese bereits abdeckt.

Abhängig davon wie neue Softwares bei diesen Simulationen abschneiden werden sie anschließend dem Fahrzeughalter vorgeschlagen oder auch nicht. Durch diesen Prozess wird nicht nur die Kundenzufriedenheit gesteigert, sondern auch mehrere Nutzenversprechen erfüllt bzw. erarbeitet (*NV-3, NV-4, NV-5*).

Die in Kapitel 2.2.3 vorgestellte Möglichkeit der direkten Suche anhand einer Situation sollte im übrigen auch bedacht werden. Sie kann ebenfalls automatisiert werden, indem das Fahrzeug automatische Suchanfragen an den Server verschickt, sobald eine nicht bewältigebare Situation auftritt.

Angebots Unterbreitung

Die Angebots Unterbreitung hat zwei wichtige Bestandteile: Zum einen die Erstellung eines Angebots unter Beachtung der in Kapitel 2.3.3 dargestellten Prinzipien. Zum anderen ist der Zeitpunkt, zu dem die Software dem Fahrer vorgeschlagen wird geeignet gewählt sein. Hierfür kann der in Kapitel 2.3.3 vorgestellte SUPR implementiert werden. Die Fahrzeughalter sollen die Kaufart (*i.e. Kauf, Miete, Abo*) selber auswählen können und so das Angebot zugunsten ihrer Zwecke abändern.

Bereitstellung elektronischer Zahlungsschnittstellen

Damit der Kauf von Fahrzeughaltern getätigter werden kann, muss mindestens eine gängige Zahlungsschnittstelle (*MasterCard²³, PayPal²⁴, o.A.*) zur Verfügung gestellt werden. Durch multiple Zahlungsschnittstellen kann ein größerer Kundenstamm erreicht werden. Um weitere eigene Umsätze zu generieren, kann hier auch eine eigene Schnittstelle eingefügt werden. Hierdurch kann ein weiterer Betrieb-

²³

²⁴

szweig (*Bank*) aufgebaut werden. Fahrzeughalter können zum Nutzen dieser gebracht werden, indem der Kauf mittels dieser Schnittstelle beispielsweise günstiger ist. Ohne eine elektronische Zahlungsschnittstelle ist das betreiben des Shops und somit die orts- und zeitunabhängige Bereitstellung von Software nicht möglich.

Marketing

Wie im Business Model beschrieben, ist der Umfang des Marketings maßgebend für den Erfolg des Shops. Das Marketingkonzept sollte auf möglichst vielen Kanälen zu sehen sein und dabei möglichst viele Alters- und Kundengruppen ansprechen. Es können Studien herangezogen werden, wie der Google Playstore oder andere vergleichbare Plattformen sich entwickelt haben. Außerdem ist es ratsam "Markenbotschafter" zu finden, die Content für Social Media Plattformen entwerfen. Coca Cola hat dies beispielsweise erfolgreich mit Coke TV²⁵ geschafft.

Im Anschluss an das Vermarkten und Vertreiben von Software muss diese auch "ausgeliefert" werden. Die in diesem Aspekt anfallenden Bausteine werden dem Vierten Schritt einer Wertschöpfungskette, der **Ausgangslogistik** zugeordnet.

Gesicherter Download über das Internet

Die Bereitstellung von Software wird mit dem Download abgeschlossen. Dieser Download soll zum einen orts- und zeitunabhängig sein, außerdem muss die Download Schnittstelle sicher sein. Um ersteres zu gewährleisten, sollte der 5G-Ausbau zusammen mit den Mobilfunkanbietern und anderen gefördert werden. Außerdem ist eine Kooperation mit Mobilfunkanbietern denkbar, da eine Vielzahl von Autos künftig große Mengen an Daten verschicken werden. Dadurch könnten für die Fahrzeughalter oder auch für Shopbetreiber immense Kosten entstehen. Durch eine Partnerschaft können die Kosten skalierbar gehalten werden, indem Mobilfunkanbieter, Shopbetreiber und Automobilhersteller zugunsten der Kundenzufriedenheit kooperieren.

Damit die langfristige Zufriedenheit von Kunden sichergestellt werden kann, müssen auch nach dem Kauf für den Kunden wertschöpfende Aktivitäten durchgeführt werden. Diese werden dem letzten Schritt der Wertschöpfungskette, dem **Kundendienst**, zugeordnet.

Verwaltung von gekaufter Software

Damit Fahrzeughalter wissen, welche Softwares sie gekauft, geliehen oder gemietet haben soll für sie die eigenständige Verwaltung von Software möglich sein. Softwares sollen auch "deaktiviert" werden. Hierdurch behalten Fahrzeughalter im übrigen auch den Überblick.

Überwachung installierter Software

Um Fahrzeughaltern vor Augen zu führen, welche Softwares nötig für das Fahrzeug sind und welche nicht, sollten diese Einblick in eine Art Überwachung der Software erhalten. Hier können Grundlegende Statistiken geführt werden wie

- die durchschnittliche Nutzungsdauer der Software in Prozent
- das Datum, an welchem die Software das letzte mal genutzt wurde

²⁵text

- oder eine Liste, in welcher zu sehen ist welche Softwares an welchem Tag der Woche vermehrt genutzt werden.

Aufgrund der steigenden Bedeutung des Datenschutzes könnte auch eine Ansicht erteilt werden, welche zeigt welche Daten von Softwares erhoben werden. Hierdurch könnte das Vertrauen von Kunden gestärkt werden (*Oder auch nicht, je nach dem welche und wie viele Daten erhoben und gespeichert werden*).

Kundenservice und Beratung

Um vor allem die Bindung zu älteren Kunden aufzubauen, kann das einrichten einer Telefonischen Beratung für große Softwarekäufe gut sein. Bei der Ersteinrichtung eines Fahrzeugs können Fahrzeughalter hierdurch unterstützt werden. Auch ein klassischer Kundenservice kann gut sein. Dieser Kundenservice muss nicht telefonisch erfolgen, sondern reicht auch in Form von Kommentare, Bewertungen und Feedback im Hinblick einer Software oder des Shops im generellen.

Überblick

Wird Software von Software Providern beim Softwareshop eingereicht, durchläuft sie die Fünf Schritte der Wertschöpfungskette. Jeder einzelne trägt dazu bei, das Fahrzeughalter am Ende selbstständig genau die Software installieren können, die sie benötigen.

1. Eingangslogistik

Anfangs muss die Sicherheit von Software verifiziert werden und es müssen Metadaten und Kennwerte für die Software festgelegt werden. Beides unterstützt die Nutzung des Shops.

2. Operationen

Ist die Software bereit für die Aufnahme in den Shop, muss sie dort eingepflegt werden. Innerbetrieblich hergestellte Software wird ebenfalls dem Shop hinzugefügt. Dieser wird immer weiterentwickelt und passt sich den Anforderungen der Nutzer an.

3. Marketing und Vertrieb

Der Vertrieb von Software wird durch die automatische Erkennung eines Softwarebedarfs gekennzeichnet. Kunden kriegen Vorschläge, die auf der Basis ihrer eigenen Fakten und Statistiken beruhen. Das unterbreiten eines elektronisch abschließbaren Angebots lässt die Fahrzeughalter Softwares kaufen.

4. Ausgangslogistik

Bei der Verteilung der Software muss ein sicherer Download möglich sein. Hierzu ist es notwendig, in den Ausbau von Netzwerkstrukturen zu investieren.²⁶

5. Kundendienst

Zur Steigerung der Kundentreue und -zufriedenheit ist das bereitstellen einer Vielzahl von Kundendiensten notwendig. Diese ermöglichen unter anderem das Verwalten und Überwachen installierter Softwares.

Während all diese Schritte durchlaufen werden, geben die *unterstützenden Aktivitäten* stetig Mehrwerte zur Produktion hinzu. Sie sind also *indirekt am Erfolg beteiligt*. Diese werden im folgenden vorgestellt und ihre Rolle im Ablauf der Wertschöpfungskette verdeutlicht.

²⁶Quelle

3.2.2 Unterstützende Aktivitäten

Es gibt Vier Unterstützende Aktivitäten. So ist eine sinnvolle **Unternehmensinfrastruktur** zu schaffen, mit guter Anbindung zur Außenwelt, schnellem Internet und Mitarbeiter freundlichen Büroräumen, in welchen auch Freizeitaktivitäten möglich sind. Weiterhin gibt es auch die **Personalwirtschaft**, oder auch Human-Resource-Management. Diese beschreiben wichtige, die Mitarbeiter unterstützende Strukturen und Anhaltspunkte.

Weiterbildungen

Die kontinuierliche Weiterbildung der Entwickler, so zum Beispiel das gemeinsame lernen neuer Programmiersprachen & -frameworks, oder auch gemeinsame Entwicklung des eigenen Lifestyles. Weiterbildungen können **alles** sein, was die Mitarbeiter voran bringt. Nicht nur Aspekte des Arbeitsgebiets sondern auch persönliche Skills tragen maßgebend dazu bei, dass Mitarbeiter aufgrund höherer Zufriedenheit besser arbeiten.²⁷ Das gemeinsame Lernen einer neuen Programmiersprache für die Back-End Entwicklung unterstützt direkt die (Weiter-)Entwicklung des Shops, welche im Kontext der Operationen durchgeführt wird.

Im Kontext eines Software Shops ist die stetige **Technologieentwicklung** wegweisend für den Erfolg des Unternehmens.²⁸ Hierzu gehört nicht nur die Schaffung von Werten für die innerbetriebliche Wertschöpfungskette, sondern auch die Supply Chain übergreifend Unterstützung der Partner und Kunden.

IDE Entwicklung

Das entwickeln einer eigenen Entwicklungsumgebung (*IDE*) für die Entwicklung von Software kann sowohl die eigene Softwareentwicklung als auch die anderer Software provider unterstützen. Mit designierten Simulatoren und integrierten Sicherheitschecks könnte der Weg von Idee über Entwicklung hin zu Bereitstellung im Shop verkürzt werden. Diese sollte immer weiterentwickelt werden und neue Funktionen erhalten, die den Entwicklungsvorgang unterstützen oder sogar beschleunigen. Ein Vergleich aus der Realität ist hier Android Studio aber auch die Eclipse IDE, welche in Zusammenarbeit mit Unternehmen aus der ganzen Welt entwickelt wird.

Dokumentation und API-Reference

Die Entwicklung von Software wird zusätzlich von der Bereitstellung einer ausführlichen API-Reference und einer technischen Dokumentation (mit Beispielen) positiv unterstützt. Die API-Reference sollte alle Funktionen von Klassen ausführlich beschreiben und im Nutzungskontext vorstellen. Diagramme unterstützen zusätzlich das Verständnis der API. Eine technische Dokumentation kann sehr vielfältig sein. Um die Entwicklung neuer Softwares voranzutreiben ist es Sinnvoll eine Einführung ('*Getting Started Guide*') in die Entwicklung von Softwares für ein Fahrzeug zu geben. Google stellt in diesem Kontext das *Android Jetpack*²⁹ sowie weitere Dokumentationen mit Anleitungen und Videos zur Verfügung.³⁰. Auch und vor allem sollten hier die Sicherheitskriterien welche im Kontext der Wertschöpfungskette in Abschnitt 3.2.1 erwähnt wurden vorgestellt werden. Tipps und Tricks und zu beachtende Dinge sollten hier zu finden sein.

²⁷ quelle

²⁸ quelle

²⁹ <https://developer.android.com/jetpack>

³⁰ <https://developer.android.com/guide>

Entwicklung von Analyse-Tools

Um Statistiken bzgl. der Softwarenutzung überblicken zu können, sollten Analyse-Tools entwickelt werden. Dank einer Web-App, über welche Software Provider generelle (*Nutzung in H je Tag*) und auch spezifischere (*Button-Clicks*) Statistiken ihrer Softwares sehen können, wären Entwickler dazu in der Lage ihre Software zu verbessern.

API-Entwicklung

Neben einer ausführlichen API-Reference, muss die API selber stets erweitert werden. Hier gibt es zum einen die Software-API. Über diese müssen Systeme des Fahrzeugs angesteuert werden können und im generellen auch festgelegt werden wie die Software angesteuert wird. Die Weiterentwicklung der Software-API kann das ansteuern neuer Automarken oder die Anbindung an eine neue Schnittstelle umfassen.

Die letzte unterstützende Aktivität ist die **Beschaffung**. Ihre Aufgabe ist, dass alle Mitarbeiter mit den nötigen Materialien, das heißt Computern, Tastaturen, Stiften oder anderem versorgt werden.

3.3 Zusammenfassung

Um die wirtschaftliche Perspektive abzuschließen, werden im folgenden die Ergebnisse des Kapitels zusammengeführt. Das Business Model Canvas hat einen weitreichenden Blick auf den Software Shop geboten. Anhand der Kundensegmente konnten Aspekte der Kundenbeziehungen und des Marketings identifiziert werden. Vor allem letzteres bedarf weiterer Arbeit. Ein gut geplantes und sinnvolles Marketingkonzept ist wichtig für den Erfolg des Shops. Auch die Nutzenversprechen wurden anhand der Kunden abgeleitet. Aus ihnen werden die Einnahmequellen, als auch die Schlüsselressourcen und -aktivitäten abgeleitet. Die Identifikation der Schlüsselpartner und das darstellen der Kostenstrukturen runden den grundlegenden Blick auf die Aufgaben eines neuen Markts ab.

Die Schlüsselaktivitäten des Businessmodels konnten im Rahmen einer Wertschöpfungskette logisch unterteilt und erläutert werden. Hierdurch wurde zum einen der Ablauf verdeutlicht durch welchen eine neue Software zur Aufnahme in den Shop gehen musste, zum anderen wurden auch die nur indirekt mit der Bereitstellung in Verknüpfung stehenden Aufgaben erläutert. Die Wertschöpfungskette, welche die Aufgaben des Software Shops definiert, ist in der Supply Chain als **Software Shop** dargestellt. Diese zeigt die anderen an der Wertschöpfung beteiligten Schlüsselpartner und in welcher Beziehung diese zueinander stehen.

Um den Ablauf eines Einkaufs im Prototypen darzustellen, werden im folgenden technische Konzepte vorgestellt, welche in diesem implementiert werden können. Sie verdeutlichen zum einen den Kommunikationsablauf zwischen Auto und Service Providern im Kontext einer autonomen Parkplatznutzung. Außerdem wird ein Klassifizierungskonzept von Software vorgestellt, welches im Kontext der Eingangslogistik (*Software Klassifizierung*) genutzt werden kann.

4 Technische Konzepte

Im vorherigen Kapitel wurden viele Aufgaben aufgedeckt, die im Kontext eines Software Shops entstehen. Einige dieser Aufgaben, wie die Sicherheitsverifikation und die Klassifizierung von Software benötigen zur möglichen Bearbeitung Konzepte zur Bewertung. Hierzu wird zunächst ein mögliches Klassifikationskonzept für Software vorgestellt. In diesem werden Berechtigungen und Kennzahlen von Softwares vorgestellt. Durch das Klassifizieren von Software können bei der Suche nach Software bereitgestellte Filter genutzt werden, welche Kunden bei der Suche der gewünschten Software unterstützt. Im Anschluss daran werden in Kapitel 4.2 drei Kommunikationskonzepte vorgestellt, welche den Ablauf der Kommunikation zwischen Fahrzeug, Server und anderen Akteuren der Umwelt im Kontext bestimmter Anwendungsfälle darstellen.

4.1 Klassifizierung von Software

Das Ziel der Software Klassifikation ist es, dem Kunden beim Kauf unterstützende Kennzahlen und Gruppierungen zur Verfügung zu stellen. Anhand dieser können sinnvollere Kaufentscheidungen getätigt werden und Kunden erhalten potentiell einen größeren Wert von der gekauften Software, wodurch die Wahrscheinlichkeit eines weiteren Kaufs steigen kann.³¹

4.1.1 Berechtigungen und Details

Eine dieser Kennzahlen ist die Anzahl genutzter Berechtigungen des Fahrzeugs. Nicht nur die Anzahl, sondern auch der Umfang einer Berechtigung können Kunden vom Kauf abschrecken.³² Eine Berechtigung muss der Software vom Fahrer erteilt worden sein, um die entsprechenden Dienste ausführen zu können. Eine Berechtigung erteilt der Software den Zugriff auf mögliche Systeme des Fahrzeugs. Einige Softwares können daher nicht genutzt werden, wenn ihnen nicht alle benötigten Berechtigungen erteilt wurden. Im Kontext intelligenter Fahrzeuge können mögliche Berechtigungen von der Architektur dieser abgeleitet werden, welche in Abbildung 9 dargestellt wird. Berechtigungen werden von den Entwicklern zur Software hinzugefügt und sind somit direkt zum auslesen im Shop verfügbar.

Berechtigungen können in zwei Gruppen unterteilt werden. "*Normale*" Berechtigungen benötigen keiner direkten Zustimmung der Fahrzeughalter, da ihre alleinige Nutzung ungefährlich ist. "*Kritische*" Berechtigungen sind solche, die im direkten Bezug zur Privatsphäre des Fahrzeughalters stehen. Für diese müssen Fahrzeughalter den Berechtigungen direkt zustimmen. Jede Berechtigung erlaubt es der Software bestimmte weitere Daten des Fahrzeugs zu lesen oder zu beeinflussen.

Normale Berechtigungen

READ _ PERCEPTION

READ _ PREDICTIONS

READ _ CONTROLS

READ _ ACTUATOR _ X

³¹quelle

³²quelle: android buch(1)

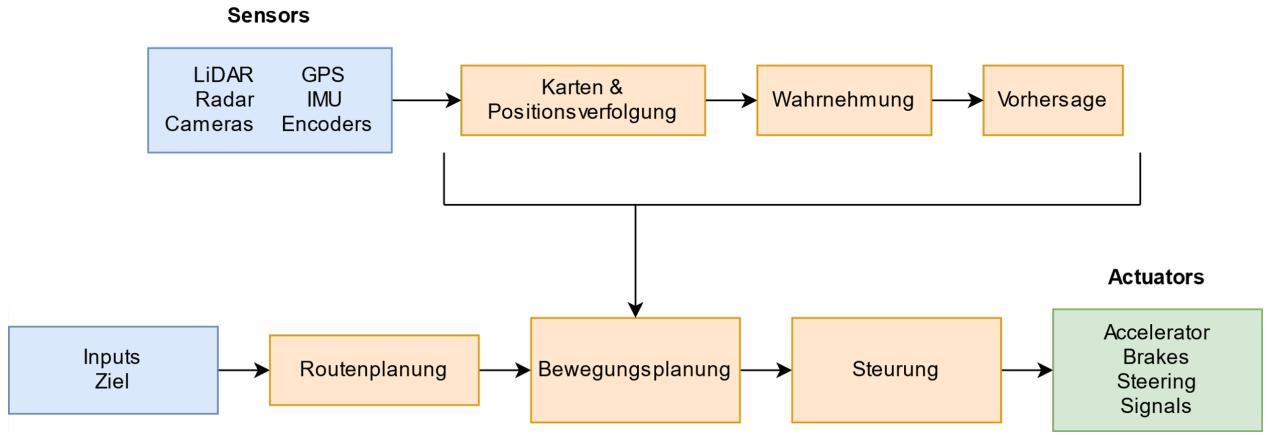


Abbildung 9: Jeff Schneider: Architecture of Autonomous Vehicles

Kritische Berechtigungen

```

READ_SENSOR_X
READ_TRACKING
READ_ROUTE_PLANNING
READ_MOVEMENT_PLANNING
READ_KAMERA_X
INPUT_NEW_GOAL
FEED_TRACKING
FEED_PERCEPTION
FEED_PREDICTIONS
FEED_ROUTE_PLANNING
FEED_MOVEMENT_PLANNING
FEED_CONTROLS

```

Weitere Systeme/services des Fahrzeugs? Kommunikation mit anderen Akteuren
 Zugriff auf Internet: Für datenbanbkbackups nötig
 APK installieren dürfen

- A. name
- B. Berechtigung erklären (wovon abgeleitet, was ist der use-case)
- C. API und Sinn der API

4.1.2 Softwarekennzahlen

Softwarekennzahlen sind Zahlenwerte, welche auf Basis der Software und ihrer allgemeinen Nutzung ermittelt werden. Sie sind im Shop klar Sichtbar und sollen den Fahrzeughalter bei der Kaufentscheidung unterstützen. In der Regel gilt, je höher eine Klassifizierung desto besser. Das ermitteln der Softwarekennzahlen erfolgt entweder im Rahmen der Eingangslogistik im Klassifizierungs Baustein oder sie werden anhand regelmäßiger Nutzungsanalysen der Software aktualisiert.

Softwarebasierte Kennzahlen

Jeder Software können vom Entwickler OpenScenario-Dateien beigefügt werden. Im Rahmen einer Simulation sollen Fahrzeuge mit unterschiedlich viel installierten Softwares diese Situation durchfahren. Dabei soll ein Grad der Fahrzeugschonung berechnet werden, welche aussagt ob und wie stark bestimmt Teile des Fahrzeugs beansprucht werden. Diesen Fahrzeugen wird anschließend die neue Software hinzugefügt, die Szenarien werden erneut durchfahren und die Werte werden nochmals gemessen. Durch einen Vergleich beider Werte kann so beispielsweise festgestellt werden, dass ohne die neue Software der Abnutzungsgrad des vorderen rechten Rades höher ist als mit der neuen Software.

Ob die Bestimmung dieser Werte möglich geschweige denn Aussagekräftig ist, wurde in dieser Arbeit nicht geprüft. Nichtsdestotrotz soll dies stellvertretend für möglich zu bestimmende Kennzahlen stehen, die im Kontext intelligenter berechnet werden können, wie dem Sicherheitsgrad der Software in den für sie vorgesehenen Situationen.

Nutzungsbasierte Kennzahlen

Jedes Fahrzeug zeichnet Statistiken über die installierten Softwares auf. Diese werden den jeweiligen Fahrzeughaltern im Shop präsentiert. Um andere Fahrzeughalter beim Kauf zu unterstützen, werden die aufgezeichneten Statistiken in festen Zeitintervallen an den Shop geschickt. Dieser sortiert die aufgenommenen Statistiken nach der Region in denen sie aufgenommen wurden. Durch die Analyse von Daten können Kennzahlen wie die "*Durchschnittlich gewonnene Zeit je Woche*" oder die "*Durchschnittliche Anzahl an Interaktionen mit anderen Akteuren pro Woche*" erfasst werden.

Bewertung des Software von anderen Fahrern

Neben Kennzahlen, die auf Basis von Zahlen und Fakten bestimmt worden sind, sollt in jedem Fall eine Bewertungsmöglichkeit einzelner Softwares möglich sein. Eine Bewertung liegt zwischen ein bis fünf Sternen und kann optional einen Kommentar beigefügt bekommen. Durch das Lesen von Bewertungen anderer Fahrzeughalter können potentielle Kunden einen Einblick in den tatsächlichen Nutzen der Software haben.

Die vorgestellten Softwareberechtigungen und -Kennzahlen werden im Shop zu jeder Software angezeigt. Hierdurch kann der Fahrzeughalter auf einen Blick einen ersten Eindruck der Software kriegen. Neben der Darstellung im Shop können Kennzahlen und Berechtigungen auch als Such-Filter im Shop verwendet werden. Kunden können die Suche nach passenden Softwares hierdurch entsprechend ihrer Wünsche anpassen. Weitere Filter sind der Preis einer Software oder der Name. Durch weitere Klassifizierungen können Softwares möglicherweise auch Kategorisiert werden, was die Fahrzeughalter bei der Suche unterstützt.

4.2 Kommunikationsprotokoll

Damit Software heruntergeladen und anschließend genutzt werden kann, müssen das Fahrzeug, der Software Shop und auch potentiell weitere Akteure des Straßenverkehr (*Autos, Ampeln, Service Provider, oA.*) miteinander kommunizieren. Der Ablauf dieser Kommunikation muss sicher sein. Zu dieser Sicherheit können Features aus Uptane(2.3.1) sinnvoll genutzt werden.

Das Fahrzeug Manifest liefert Informationen darüber, welche Version welcher Softwares auf einem Fahrzeug installiert sind. Hierdurch und durch das in Kapitel 2.2.2 vorgestellte OpenScenario Speicherformat können komplexe Informationen einfach verschickt werden. Die entworfenen Protokolle beschreiben die Abfolge der einzelnen Schritte, welche zur Erreichung des Ziels notwendig sind. Jedes einzelne wird mit der Intention entwickelt, sie im Prototypen zu implementieren. Sie sollen möglicherweise sichere und nutzerfreundliche Kommunikationsabläufe in gewissen Situationen des Alltags darstellen.

4.2.1 Kommunikationsprotokoll: Selbstständiger Softwarekauf

Das in Abbildung 10 dargestellte Protokoll zeigt einen möglichen Kommunikationsablauf, wie er stattfindet wenn ein Fahrzeughalter selbstständig Software aus dem Shop kauft. Der Fahrzeughalter kann im Shop nach Softwares suchen, indem der Name einer Software gesucht wird oder aber Suchfilter verwendet werden. Wurde eine Software ausgewählt (1), wird eine Installationsanfrage für diese an den Server geschickt. Der Server bereitet ein **Installationspaket** für die neue Software vor (2), in welchem eine neue Version des Manifests enthalten ist, die neue Software sowie ein entsprechendes preisliches Angebot. Das Paket wird an das Fahrzeug zurück geschickt und in Folge dessen muss der Fahrzeughalter eine Kaufentscheidung treffen (3). Die Eingabe der Entscheidung erfolgt über die Mensch-Maschine-Schnittstelle und je nach Entscheidung, wird die Software installiert oder nicht.

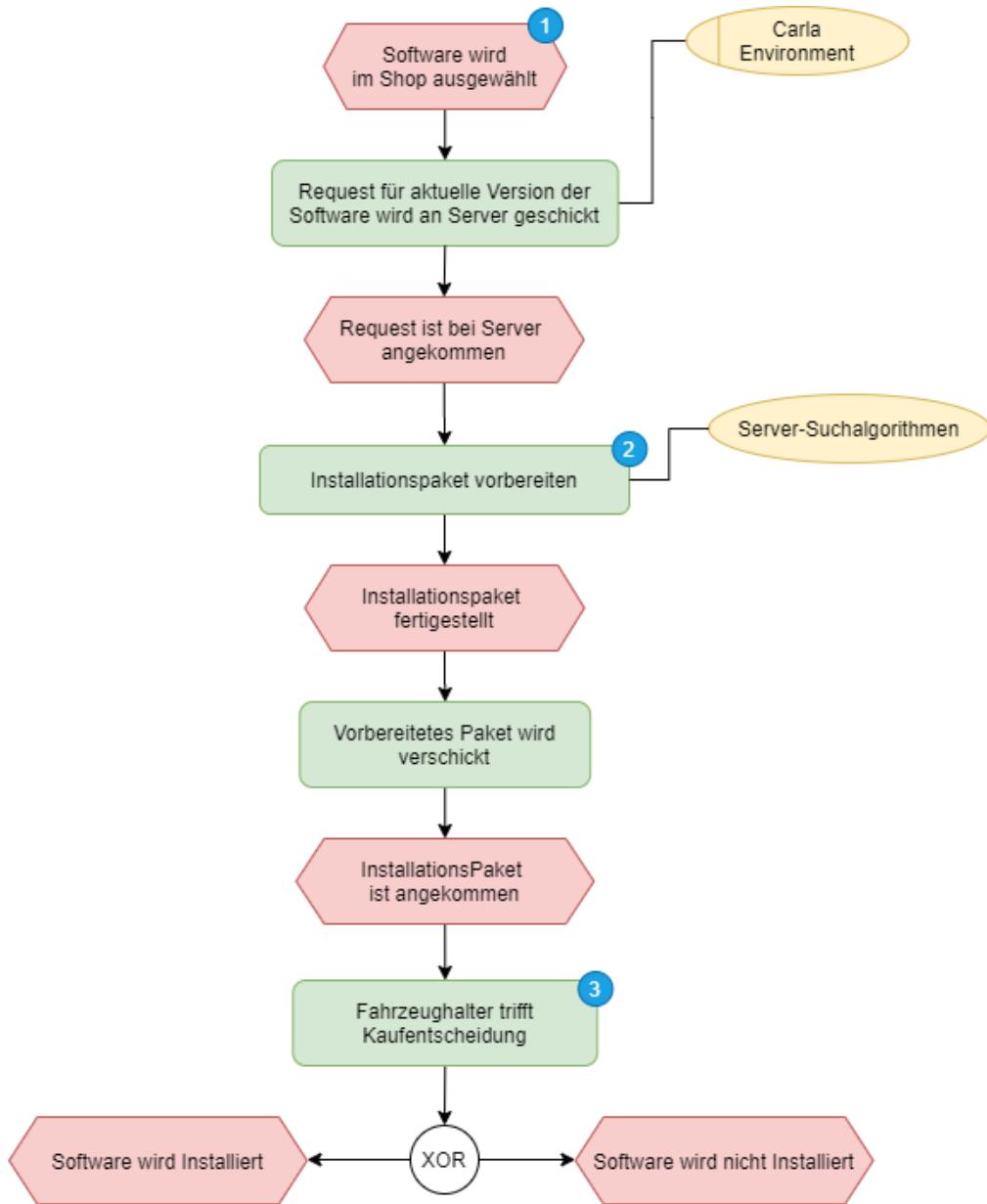


Abbildung 10: Eigenständige Installation von Software

4.2.2 Kommunikationsprotokoll: Installationsvorschlag von Service Provider

Abbildung 11 stellt den Ablauf einer Service Registrierung dar. Stellen Sie sich folgende Situation vor: Sie sitzen in Ihrem Fahrzeug und möchten in die Innenstadt. Das Fahrzeug schlägt Ihnen vor einen Parkplatz nahe der Fußgängerzone anzusteuern. Das Fahrzeug fährt dort hin ohne zu wissen, dass für die Nutzung des Parkplatzes eine Software notwendig ist, welche die Kommunikation mit dem Parkautomaten ermöglicht. Das Protokoll setzt ein, sobald das Fahrzeug am Parkplatz ankommt. Es fährt in eine Zone die vom Parkautomaten überwacht wird (1). Wird ein Fahrzeug erkannt, versucht der Parkautomat sich beim Fahrzeug zu registrieren. Ist die Nachricht im Fahrzeug angekommen wird überprüft, ob bereits eine Software auf dem Fahrzeug

installiert ist welche die Nachrichten verarbeiten kann (2). Dies wird anhand der Software ID gemacht. Ist die Software bereits installiert, wird überprüft ob die installierte Version der Software aktuell ist. Wenn ja, kann der Service genutzt werden. Dieser Ablauf wird im kommenden Kapitel vorgestellt. Ist die Software noch nicht installiert oder die installierte Version ist nicht mehr aktuell wird der Fahrzeughalter gefragt, ob er die Software aktualisieren bzw. kaufen möchte (4). Dies geschieht über eine Eingabe in der Mensch-Maschine-Schnittstelle. Wird diese Anfrage abgelehnt, wird keine Software installiert und das Fahrzeug verlässt die Registrierungszone wieder. Andernfalls sendet das Fahrzeug eine Installationsanfrage an den Server, welche das Manifest des Fahrzeugs und Informationen zur Software beinhaltet. Der Server nimmt die Anfrage an und erstellt ein Installationspaket für die entsprechende Software (5). Ist das Paket fertig, wird es an das Fahrzeug verschickt. Handelt es sich um ein Update einer bereits installierten Software, trifft der Fahrer keine Kaufentscheidung sondern akzeptiert die neuen Bedingungen der Software. Haben sich diese nicht geändert, wird das Update automatisch durchgeführt. Muss die Software neu gekauft werden, trifft der Fahrzeughalter eine Kaufentscheidung. Er kann das ihm vorgestellte Angebot entsprechend seiner Bedürfnisse für die Software anpassen (*Leihe statt Kauf, etc.*).

4.2.3 Kommunikationsprotokoll: Nutzung eines Service

Das letzte Kommunikationsprotokoll wird genutzt, wenn ein Service vom Fahrzeughalter genutzt wird. Dies passiert **immer** direkt nach dem vorherigen Kommunikationsprotokoll, wenn die installierte Software aktuell ist (1). Der Service wird dem Fahrzeughalter über die Mensch-Maschine-Schnittstelle angezeigt (2) und der Fahrer muss entscheiden, ob er diesen nutzen möchte oder nicht. Die Antwort wird an den Parkautomaten gesendet (3), welcher die Entscheidung des Fahrers auswertet. Wurde der Service abgelehnt, wird eine Abschiedsnachricht zusammen mit einer Aufforderung das Gelände zu verlassen an das Fahrzeug gesendet. Wurde der Service angenommen, beinhaltet die zurückgeschickte Nachricht einen zugewiesenen Parkplatz und ein Parkticket. Das Fahrzeug verarbeitet die Nachricht (4), zeigt diese auf der Mensch-Maschine-Schnittstelle an und steuert den Wagen entsprechend der Nachricht.

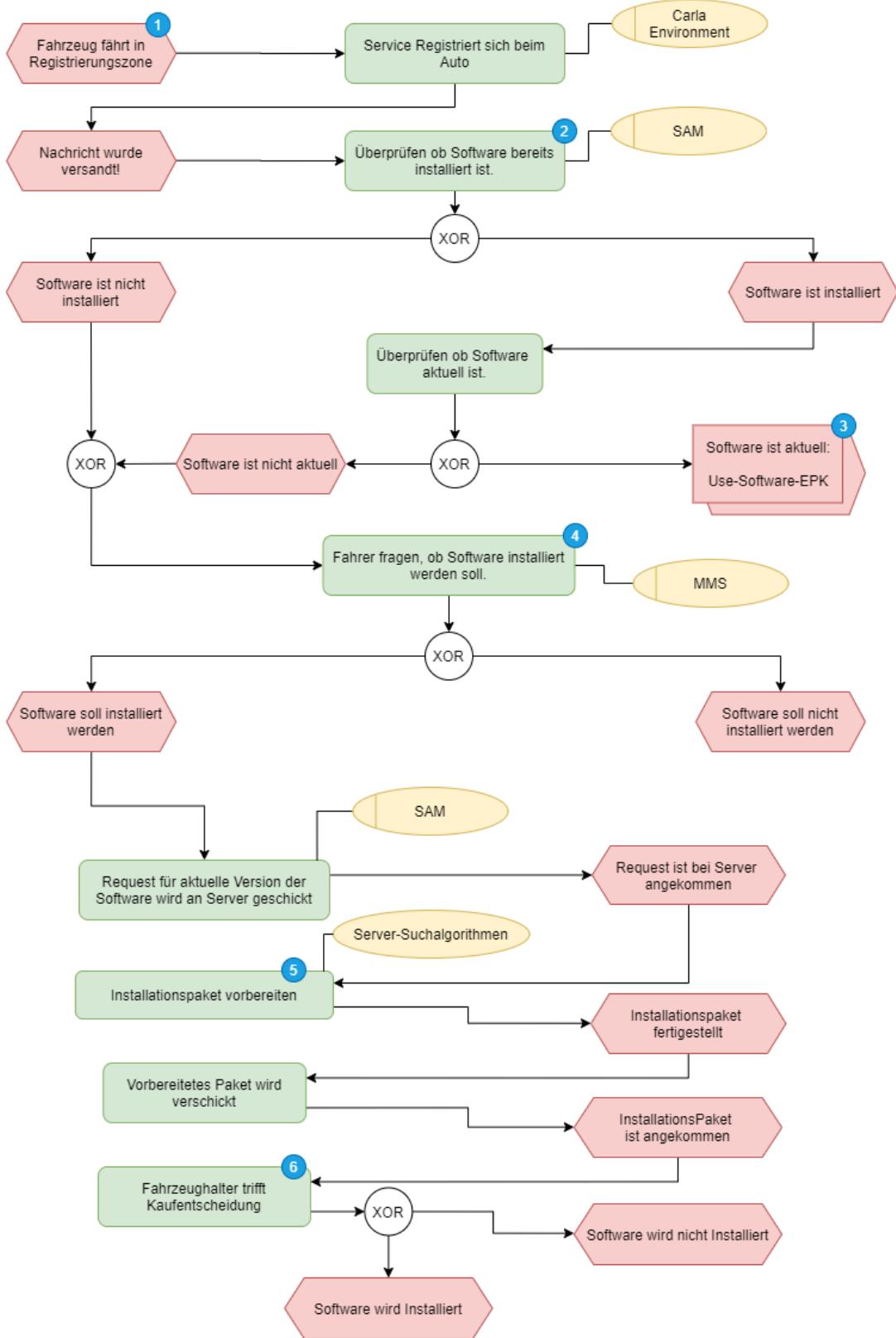


Abbildung 11: Installationsprozess von Software

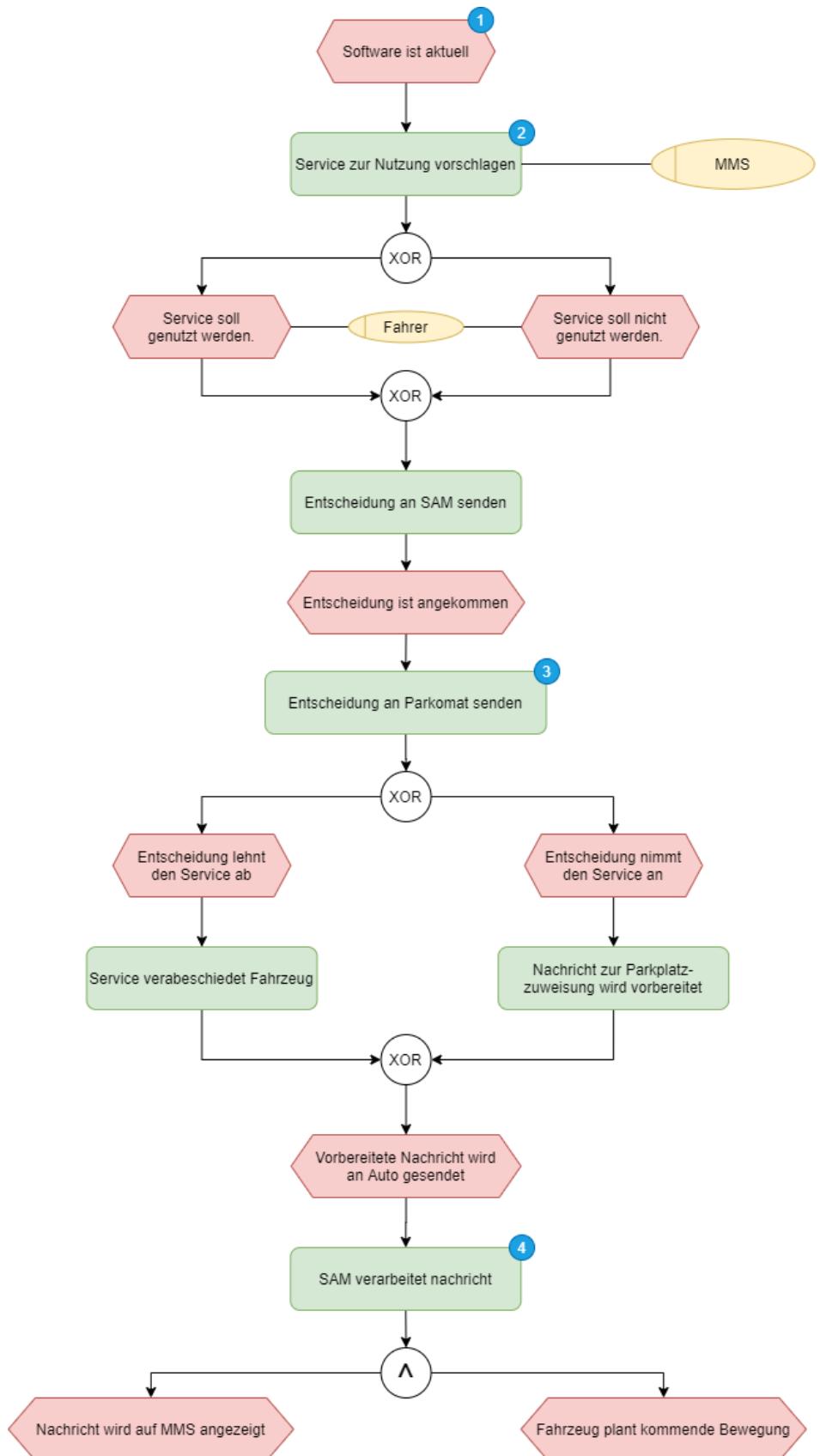


Abbildung 12: Nutzungsprozess von Software ⁴⁶

4.3 Ausblick

Die in diesem Kapitel erstellten Konzepte wurden mit der Intention entwickelt, einzelne Bausteine der Wertschöpfungskette zu unterstützen. In Kapitel 3.2 wurden weitere Konzepte und notwendige Arbeitsvorgänge vorgeschlagen, für welche ebenfalls technische oder zumindest theoretische Konzepte zu erstellen sind.

Im Hinblick auf den Prototypen wurden Kommunikationsprotokolle erstellt, welche für die Einbindung in eine Uptane-basierte Fahrzeug-Server-Kommunikation anwendbar sind. Der Ablauf der Protokolle, die im Forschungsseminar erarbeitete Softwarearchitektur und auch weitere in der Arbeit gewonnene Erkenntnisse und Ideen werden im folgend vorgestellten Prototypen visuell veranschaulicht werden.

5 Vorstellung des Prototypen

Der Prototyp demonstriert, wie eine Software entsprechend der in der Arbeit identifizierten Bausteine einem Fahrzeughalter bereitgestellt werden kann. Konkret wird die Bereitstellung einer Software dargestellt, mit welcher **ein Fahrzeug selbstständig ein Parkticket kaufen und anschließend zum Parkplatz fahren kann**. Zunächst wird in Kapitel 5.1 dargestellt, wie der Prototyp installiert werden kann. Im Anschluss daran wird in Kapitel 5.2 die grobe Architektur des Prototypen vorgestellt. Darauffolgend werden zwei wichtige Einzelsysteme des Prototypen, der Server und der Software Applikation Manager, im Detail betrachtet. Im Anschluss an die Vorstellung der Architektur erfolgt Darstellung der Funktionen des Prototypen und wie diese genutzt werden können.

5.1 Architektur des Prototypen

Abbildung 13 zeigt die Grundlegende Architektur des Prototypen. Über den OEM-Server können Software Heruntergeladen werden. Der **Software Applikation Manager (SAM)**, die **Carla Simulation** und die **Mensch Maschine Schnittstelle (MMS)** bilden gemeinsam die Repräsentation des Fahrzeugs. Die Python-basierte Carla Simulation stellt das Fahrzeug in einer Stadt dar. Dieses Fahrzeug ist die visuelle Repräsentation des Prototypen. Im Kontext einer *Model-View-Controller (MVC)*³³ Architektur ist sie demnach als View zu sehen. Dies restlichen Module sind alle in Java geschrieben. Die Mensch Maschine Schnittstelle ist ein Android-Tablet, auf welchem eine Version des Software Shops als App installiert ist. Über diese App

³³MVC architektur quelle

können Nachrichten von dem Software Applikation Manager empfangen und verarbeitet werden. Über die App können Nutzereingaben getätigt werden um mit dem restlichen Komponenten des Fahrzeugs zu kommunizieren. Der Software Applikation Manager ist mit allen Modulen des Prototypen verbunden und stellt daher die zentrale Steuerungseinheit des Prototypen dar. Neue Software wird hier installiert und genutzt.

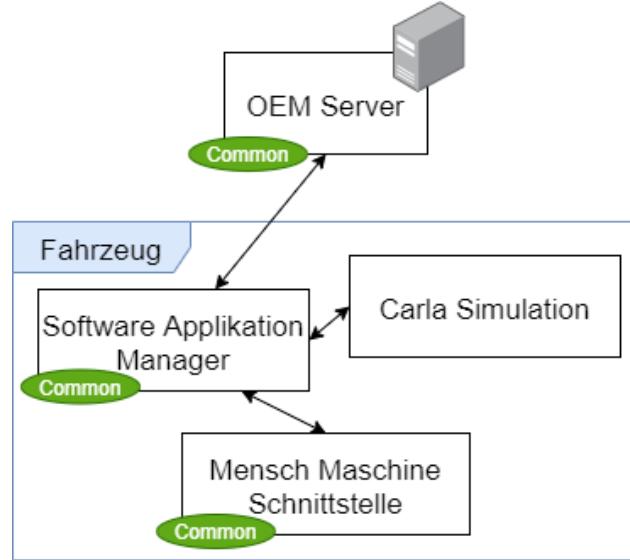


Abbildung 13: Grundlegende Architektur der Prototypen

Der Server, SAM und die MMS kommunizieren alle auf Basis der **Common-Objekte**. Die Common-Objekte sind eine Java-Library, in welcher Objekte gesammelt sind die jedes einzelne Modul kennen muss. Sie enthalten Primitive Werte (*Integer, String, Long, etc.*) und leiten alle vom Serializable-Interface ab. Abbildung 14 stellt die generelle Aufteilung der Objekte in zwei Gruppen dar.

Environment-Objekte sind Objekte, welche zur Umwelt des Prototypen gehören. Damit der Server spezifische Softwares verschicken kann, wird ihm die abstrakte Klasse Software zur Verfügung gestellt als auch eine Sammlung von Softwares. Jede Software des Prototypen benötigt eine eindeutige ID und eine Methode *handleMessage()*, welche bestimmte Messages bearbeiten kann. Auch der Software Applikation Manager nutzt diese Objekte, um so die Softwares *installieren und nutzen* zu können. Weiterhin gibt es auch die Description und den Provider. Die Description wird genutzt, um Softwares oder Services, die im Prototypen implementiert werden zu beschreiben. Das Provider-Objekt tut eben dies für die Beschreibung des Anbieters der Software bzw. des Services. Das Angebot repräsentiert die Auswahlmöglichkeiten, die dem Fahrzeughalter beim Kauf einer Software über die MMS vorgeschlagen werden (*Kauf, Leihe, Abo*). Alle Drei Objekte werden in der MMS genutzt um das Angebot hinreichend darzustellen.

Neben den Environment-Objekten gibt es auch noch Messages. Diese sind alle Nachrichten, die im Prototypen zwischen den einzelnen Komponenten verschickt werden und beinhalten neben Primitiven Datentypen auch Environment Objekte. Messages werden in SoftwareMessages oder ServiceMessages unterteilt. SoftwareMessages sind all jene Nachrichten, welche zur Installation oder Nutzung einer Software notwendig sind und ServiceMessages sind jene, die zur Nutzung eines Service notwendig sind.

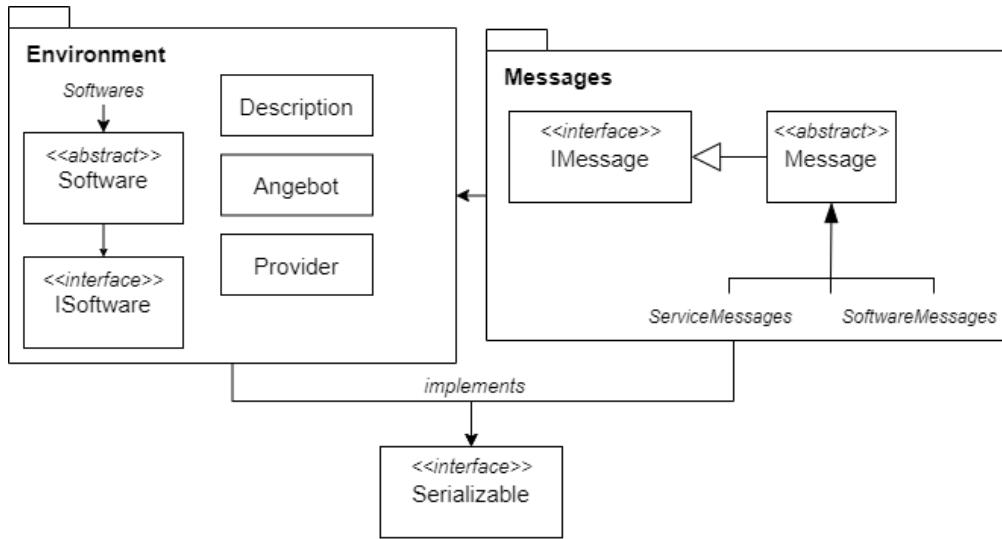


Abbildung 14: Aufbau der Common Library

5.1.1 Architektur der Software Applikation Managers (SAM)

Wie bereits erwähnt, wird über den Software Applikation Manager das Geschehen des Prototypen gesteuert. Er ist, wie in Abbildung 15 zu sehen ist, in drei kleine Module aufgeteilt. In der GUI werden alle Ereignisse des Prototypen in einem Log dargestellt. Die Log-Einträge werden vom MessageHandler eingetragen. In der "Software Control Unit" baut dieser Netzwerkverbindungen zwischen sich, dem Server, der MMS und Carla auf. Jede eingehende Nachricht wird vom MessageHandler empfangen und verarbeitet. In diesem Kontext, kann eine eingegangene Nachricht an das MMS weitergeleitet werden oder es auf eine Nutzereingabe in der GUI gewartet werden. Der Messagehandler beinhaltet einen SoftwareManager. Dieser verwaltet die auf dem "Fahrzeug" installierten Softwares und stellt diese zur Verfügung wenn sie zur Auswertung einer Nachricht benötigt werden.

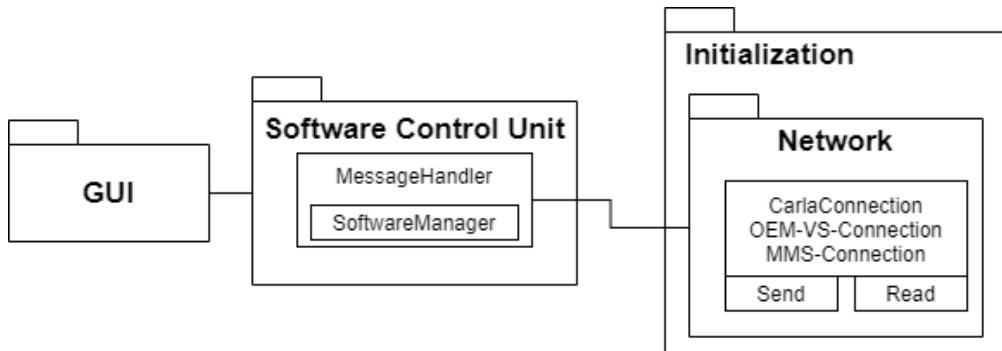


Abbildung 15: Architektur des Software Applikation Managers

5.1.2 Architektur des OEM-Servers (Server)

Die Architektur und auch die Funktionalität des Servers wurden unter der Beachtung des Uptane Standards geplant und implementiert. Die Architektur ist in Abbildung 16 dargestellt. Der Kern des Servers ist der **Director**, welcher eingehende Anfragen abarbeitet. Der Director ist mit dem **Image Repository** verbunden, in welchem sich die Softwares befinden, welche im Shop bereits bereitgestellt werden. Uptane gibt vor, eine **Inventory Database** zu führen. In dieser sind die Manifeste aller im Shop registrierter Fahrzeuge gespeichert, wie sie zuletzt von dem Director verifiziert wurden. Kommt eine Nachricht beim Director an, vergleicht dieser das in dieser enthaltene Manifest mit der in der Inventory Database enthaltenen Version. Durch den Vergleich des ByteCodes beider kann festgestellt werden, ob am Fahrzeug Änderungen von einer externen Instanz getätigten wurden oder nicht.

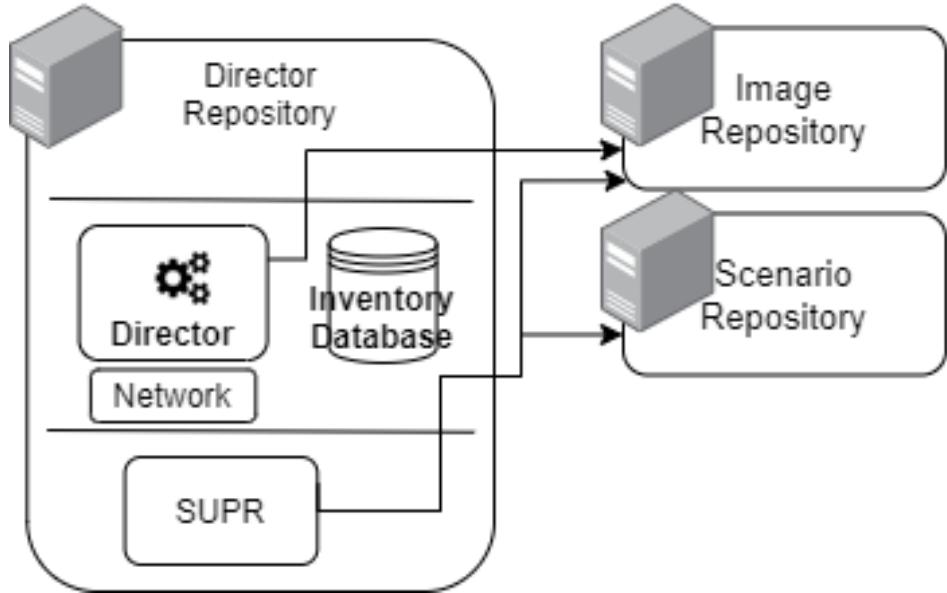


Abbildung 16: Architektur des Software Applikation Managers

Der Director kann eingehende Anfragen an den Software-User-Pattern-Recognizer (*SUPR*) weiterleiten, welcher in Kapitel zwei vorgestellt wurde. Der SUPR ist dafür zuständig, den Software-Bedarf von Fahrzeughaltern anhand ihrer Fahrdaten zu identifizieren. Er hat Zugriff auf das Image Repository und das *Scenario Repository*. In diesem werden, wie in Kapitel zwei erwähnt, die zu einer Software hinzugefügten Szenarien gespeichert. Der Director leitet Anfragen weiter, die das feststellen von Softwarebedarf umfassen. Er selber verarbeitet nur die direkten Kaufanfragen für Software.

5.2 Installation

Die Installation der einzelnen Module muss getrennt voneinander passieren. Für die vollständige Nutzung muss sichergestellt sein, dass Carla auf dem PC auf dem der Prototyp installiert werden soll lauffähig ist. Hierzu werfen Sie einen Blick in die Systemanforderungen von Carla.³⁴ Zur Not einfach einmal ausprobieren. Die einzelnen Module können auch auf getrennten Geräten genutzt werden, indem die genutzten

³⁴Carla Systemanforderungen

IP-Adressen und Ports angepasst werden. Die folgende Anleitung stellt dar, welche Schritte zur Installation der einzelnen Module getätigt werden müssen. Die Anleitung ist für Windowsgeräte ausgelegt und dauert ca.

ZEIT EINFÜGEN Minuten

5.2.1 Server und SAM Installation

1. .zip-Ordner Herunterladen

Gehen Sie auf <https://github.com/hesty98/bachelorthesis>. Klicken Sie auf den "releases"-Tab, welcher unter der Beschreibung des Repository zu finden ist. Laden Sie die Datei "prototyp.zip" herunter und entpacken Sie den Ordner auf Ihrem Laptop. In diesem Ordner finden Sie die .jar-Dateien mit welchen der Server und der SAM gestartet werden können. Der Ordner "PythonScriptsThesis" wird später für die Installation von Carla benötigt.

2. IP-Adresse aufschreiben

Um das MMS und auch Carla mit dem Software Applikation Manager zu verbinden, müssen Sie Ihre IP-Adresse kennen. Öffnen Sie hierzu Ihr Windows-Temrinal und geben Sie `ipconfig` ein. In der Zeile "IPv4-Adresse" steht Ihre IP-Adresse. Diese müssen Sie sich merken.

5.2.2 MMS Installation

1. Android Studio herunterladen

Sie haben zwei Optionen, wie die Mensch Maschine Schnittstelle genutzt werden kann. Entweder sie simulieren das Android Gerät auf Ihrem PC oder Sie installieren die App auf ein eigenes Tablet-/Smartphone. Für beides wird eine aktuelle Version von Android Studio benötigt, die Sie hier <https://developer.android.com/studio> herunterladen können. Installieren Sie Android Studio und fahren Sie anschließend vor.

2. Code importieren

Das Repository der Mensch Maschine Schnittstelle liegt auf GitHub. Über Android Studio können Sie das Projekt direkt importieren. Klicken Sie hierfür auf "File" -> "new" -> "Project from Version control" -> "GitHub" und geben Sie folgende URL ein: <https://github.com/hesty98/HMI>.

TESTEN

Alternativ kann das Projekt auch als .zip-Datei heruntergeladen werden. Der extrahierte Ordner kann in Android Studio als Projekt geöffnet werden (*File -> open -> Ordner wählen*).

3. MMS Konfigurieren

Öffnen Sie in Android Studio die Datei Connection.java und passen Sie den Host entsprechend der IP des *Software Applikation Managers* an. Die Datei befindet sich im 'network'-Ordner (*app -> java -> com.linushestermeyer.hmi -> network*).

4. MMS aufsetzen

Sie können die Mensch Maschine Schnittstelle nun entweder auf einem eigenen Gerät installieren oder

das Gerät auf Ihrem PC simulieren. Für beide Optionen Sie zuerst eine Konfiguration für die App erstellen. Folgen Sie hierzu folgender Anleitung: <https://developer.android.com/studio/run/rundebugconfig>.

4.1 Eigenes Tablet

Zur Installation auf Ihrem Tablet müssen Sie zunächst die *Entwickler-Optionen* für dieses Freischalten und *USB-Debugging* aktivieren. Folgen Sie hierfür folgender Anleitung: <https://mobilsicher.de/ratgeber/usb-debugging-aktivieren>. Wurde dies getan, kann das Gerät an den PC angeschlossen werden. Über den grünen 'Play'-Button in der oberen Menuleiste kann die App auf Ihrem Gerät installiert werden.

4.2 Virtual Device

Wenn Sie Die App **nicht** auf einem eigenen Gerät installieren wollen, müssen sie mit Android Studio ein Simulierte Gerät erstellen. Dies funktioniert nur, wenn ihr PC eine **Intel-CPUs** hat, da Android Studio den *HAXM-Service* von Intel zur Simulation nutzt. Verfügt ihr PC über eine Intel-CPUs, können Sie mit Hilfe folgender Anleitung ein Virtuelles Gerät erstelle: <https://developer.android.com/studio/run/managing-avds>. Erstellen Sie eine Instanz eines Tablets, nicht die eines Smartphone.

Haben Sie ein *VD* erstellt, können sie anschließend über den grünen 'Play'-Button in der oberen Menuleiste die App auf diesem Gerät installieren.

5.2.3 Carla Installation

1. Python installieren und SYS_PATH hinzufügen

Damit Carla auf dem PC laufen kann, muss Python in Version 3.7 auf dem PC installiert sein und dem SYSTEM_PATH hinzugefügt werden. Hierzu kann folgende Anleitung genutzt werden: <https://geek-university.com/python/add-python-to-the-windows-path/>. Hier die 3.4 nur durch eine 3.7 gedanklich ersetzen. Der PATH kann by der Installation auch durch das ankreuzen einer Checkbox automatisiert werden.

2. Carla Herunterladen

Carla kann als .zip-Ordner heruntergeladen werden. Die Dateien müssen in einen eigenen Ordner auf dem PC extrahiert werden. Als nächstes öffnen sie den Speicherort von Carla im Explorer und öffnen in diesem Fenster das Terminal (*geben sie 'cmd' in den Suchbalken des Explorers ein*). Jetzt müssen die für die Simulation notwendigen Python-Packages *pip*, *pygame* und *numpy* installiert werden. Die pip-Installation kann hier nachgelesen werden: <https://pypi.org/project/pip/>. Für die Installation von pip sollte zudem eine Adminsitratior-Konsole verwendet werden (*Windows -> suche cmd' -> 'Als Administrator öffnen'*). Ist pip installiert, können mittels folgender Zeile die übrigen packages hinzugefügt werden:

```
py -3.7 -u pip install --user pygame numpy  
EINFÜGEN HILFE
```

Im Anschluss daran sollten Sie Carla ein mal testen. Starten Sie zunächst die Carla.exe und wechseln sie im Terminal in den PythonAPI/examples Ordner (*cd PythonAPI/examples*). Führen Sie anschließend folgenden Befehl durch:

```
py - 3.7 spawn_npc.py
```

3. Skripte einfügen

In der extrahierten .zip-Datei des SAM und Servers befindet sich ein Ordner mit dem Namen *Python-ScriptsThesis*. Kopieren Sie diesen Ordner in den *PythonAPI*-Ordner von Carla und Sie haben es geschafft!

4. Carla Konfigurieren

Abschließend sollten sie noch Konfigurationen an Carla vornehmen. Zuerst wird die Stadt geändert in welcher sich das Szenario abspielen wird. Hierzu

EINFÜGEN

Haben Sie die entsprechenden Werte eingetragen, speichern und schließen Sie die Datei. Anschließend wechseln sie in den Ordner *PythonAPI/PythonScriptsThesis* und editieren Sie die `install_sw_use_service.py`. Passen Sie die IP an die des **Software Applikation Managers** an.

5.3 Nutzung des Prototypen

Die Module des Prototypen müssen in bestimmter Reihenfolge gestartet werden:

1. Server

Server.jar ausführen.

2. Software Applikation Manager

SAM.jar ausführen

3. Carla

Starten Sie Carla. Führen Sie anschließend im Terminal im Ordner "*PythonAPI/PythonScriptsThesis*" das Skript `install_sw_use_service.py` aus.

```
py -3.7 install_sw_use_service.py
```

4. Mensch Maschine Schnittstelle

Starten Sie die App auf Ihrem Gerät oder in dem Virtual Device. Gehen Sie sicher, dass die App zuvor geschlossen war.

Im folgenden haben Sie fünf Bildschirme, über die alle Module des Prototypen sichtbar sind. Die Mensch Maschine Schnittstelle wartet auf eingehende Nachrichten, ebenfalls wie Carla. Die GUI des Software Applikation Managers ist in drei Bereiche aufgeteilt. Der Log auf Linken Seite stellt alle Ereignisse aus Sicht des Fahrzeugs dar. Der Log auf der Rechten Seite der GUI stellt sämtliche Ereignisse dar, die von dem Parkautomaten wahrgenommen werden. Zwischen beiden Logs sind diverse Knöpfe, die für die Steuerung des Prototypen gedacht sind.

Auto spawnen spawnt das Fahrzeug in der Carla Simulation. Mit den WASD-Tasten und der Maus können Sie herauszoomen und das Fahrzeug suchen. Prinzipiell ist es links vor ihnen. Der Zum Parkplatz fahren-Knopf lässt das Fahrzeug in der Simulation zu einem Parkplatz fahren und es hält dort an.

Der In Erkennungsbereich fahren-Knopf lässt das Fahrzeug in den Bereich fahren, in welchem der Parkautomat dieses erkennen kann. Nun kann sich der Parkautomat über den Sende Service Registrierung-Knopf beim Fahrzeug registrieren. Der Sende ActionMessage-Knopf sendet eine Nachricht an Carla, die entweder das Auto parken lässt oder es vom Parkplatz entfernen lässt. Das Szenario kann durch den Neustarten-Knopf beliebig oft durchgeführt werden. Durch den Hack Car!-Knopf wird das Auto "gehackt". Dies soll den Fall darstellen, wenn ein Fahrzeug versucht eine Software vom Shop zu installieren aber extern Änderungen im Auto vorgenommen wurden. Durch Änderungen am Fahrzeug verändert sich das Manifest des Fahrzeug, weshalb ein Fahrzeug keine Kommunikation mit dem Server aufbauen kann. Sämtliche Ereignisse die in Folge eines Knopfdrucks passieren, werden in den Logs notiert.

Über die **Mensch Maschine Schnittstelle** kann auf vom SAM verschickte Nachrichten reagiert werden. Initial ist der Home Screen des Shops zu sehen. Erhält das die MMS eine Software- oder ServiceRegistrationMessage, erscheint ein PopUp, in welchen der entsprechende Service bzw. die Software vorgestellt wird. Es kann entschieden werden, welches Angebot genommen werden soll oder auch die komplette Transaktion mit **Cancel** abgebrochen werden.

Die Server-GUI und die Simulation stellen weitere Visuelle Repräsentationen des Prototypen dar. In beiden können die Auswirkungen nachvollzogen werden, die durch die MMS und den SAM geschehen.

5.4 Analyse des Prototypen und Ausblick auf die Weiterentwicklung

Im Prototypen wurde die im Forschungsseminar erläuterte Uptane Architektur versucht grundlegend zu implementieren. Hierdurch konnte gezeigt werden, wie ein Fahrzeug in der Zukunft von externen Eingriffen geschützt werden könnte. Die erstellte Software Architektur lässt eine einfache Erweiterung des Prototypen zu. Für einen neuen Anwendungsfall muss bloß eine neue Software erstellt werden und das Carla Skript muss angepasst werden. Dies dauert ca. zwei Stunden. Die Installation des Prototypen ist relativ Zeitaufwendig und komplex, weshalb das Bereitstellen einer detaillierte Anleitung wichtig gewesen ist.

Der Prototyp hat dargestellt, wie eine Software über eine kabellose Schnittstelle ortss- und zeitunabhängig gekauft und installiert werden kann. Zusätzlich wurde die Interaktion mit den in der Wertschöpfungskette identifizierten Service Providern implementiert, was zusätzliche Anwendungsfälle bieten kann. Über die GUIs kann nachvollzogen werden, was aktuell im Fahrzeug passiert und die Simulation kann ausreichend gesteuert werden. Die visuelle Repräsentation des Fahrzeugs in Carla unterstützt die Wirkung des Prototypen enorm, da der Betrachter direkt die Auswirkungen seiner Entscheidungen sieht. Auch die Interaktion über die Android-basierte Mensch Maschine Schnittstelle hat sich als sinnvoll erwiesen, da hierdurch ein guter Bezug zur Realität und Android Automotive gezogen werden kann.

Der Prototyp bietet noch eine Vielzahl an Erweiterungsmöglichkeiten. Der **Server** könnte um weitere Uptane-Funktionen erweitert werden. Die Erstellung eines Server-seitigen SUPR kann Türen öffnen, komplexe Suchalgorithmen auf der Basis von OpenScenario-Dateien in den Prototypen zu integrieren. Auch der **Software Applikation Manager** kann durch diverse neue Funktionen Sinnvoll erweitert werden. Ein Fahrzeug-basierter SUPR, dargestellt in Kapitel 2.3.3, kann Bestimmen von ein PopUp auf der MMS erscheinen soll und sodurch das wahllose erscheinen von PopUps verhindern. Die Steuerung über die GUI

sollte den neuen Anwendungsfällen entsprechend angepasst werden, da nicht in jedem Anwendungsfall eine Kommunikation mit einem Service Provider stattfindet. Entfernt man sämtliche Nutzereingaben des Prototypen und automatisiert so die gesamte Simulation, können die dargestellten Anwendungsfälle beliebig hoch skaliert werden, wodurch ein Shop und die Zuverlässigkeit der Bereitstellung getestet werden kann.

Im Laufe der Entwicklung wurde deutlich, dass Carla äußerst umfangreich ist. Würden die Funktionen von Carla ausgereizt werden, könnte eine Vielzahl neuer Anwendungsfälle zum Prototypen hinzugefügt und in Carla visuell dargestellt werden. Auch eine wechsel zwischen autonomen Fahrfunktionen und eigenständiger Steuerung des Fahrzeugs kann in Carla erfolgen. Im aktuellen Prototypen ist Carla nur eine visuelle Repräsentation. Durch eine Implementierung des SAM in Python kann dies korrigiert werden und die Architektur kann um ein Modul verkleinert werden. Carla wäre dadurch sowohl die visuelle Repräsentation des Fahrzeugs als auch die "Business Logik".

6 Resumé

ergebnisse zusammenfassen

schlechtes

gutes

etc

Zusätzlich können normale apps immer heruntergeladen werden. Die Besonderheit an Apps auf DriveAround ist, dass diese in das Fahrgeschehen eingreifen können.

7 Ausblick

Prototyp: Ideal zum Vorstellen auf Messen / Videos über die 'Zukunft der Automobilität'

weitere Anwendungsfälle, also erweiterung des Prototypen

SUPR als Projekt

Bedeutung User-Centered-Design im Kontext autonomer Fahrzeuge

7.1 Konzept Individuelle Darstellung von Softwares im Shop

Notiz: wird nur bei genügend Platz geschrieben, aktuell nicht fest eingeplant und nicht weiter im Text erwähnt.

Gute für Ausblick der BA bzw. des Prototypen

rehechte Asoekte des Shops

Kauf von SW für Flottenbetreiber über INternet?(?)

Eine App wo die infos zum auto sind?

References

[1] Daunis, Juergen: 'The automotive industry in transformation – business model disruption'

<https://www.ericsson.com/en/blog/2017/11/the-automotive-industry-in-transformation-business-model-disruption> (abgerufen am 14.11.2019).

[2] Logistik-info.de: 'Push und Pull Prinzip der Logistik'

<https://www.logistik-info.net/diverses/push-vs-pull-prinzip/> (abgerufen am 16.11.2019)

[3] Volkswagen: 'Modelle'

<https://www.volkswagen.de/de/modelle-und-konfigurator.html> (abgerufen am 16.11.2019)

[4] Personas QUelle definition

[5] Lex Fridman: <https://lexfridman.com/> (abgerufen am 17.11.2019)

[6] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. 2009. e DARPA urban challenge: autonomous vehicles in city traffic. Vol. 56. Springer.

- [7] Lex Fridman, Daniel E. Brown, Michael Glazer, William Angell, Spencer Dodd, Benedikt Jenik, Jack Terwilliger, Julia Kindelsberger, Li Ding, Sean Seaman, Hillary Abraham, Alea Mehler, Andrew Sipperley, Anthony Pettinato, Linda Angell, Bruce Mehler, and Bryan Reimer. 2017. MIT Autonomous Vehicle Technology Study: LargeScale Deep Learning Based Analysis of Driver Behavior and Interaction with Automation. CoRR abs/1
- [8] Kieren McCarthy. 2018. So when can you get in the first self-driving car? GM says 2019. Mobileye says 2021. Waymo says 2018. (May 2018). <https://www.theregister.co.uk/2018/05/09/first autonomous vehicles/>
- [9] Alex Davies. 2015. Oh Look, More Evidence Humans Shouldn't Be Driving. (May 2015). <https://www.wired.com/2015/05/ oh-look-evidence-humans-shouldnt-driving/> [9] Terrence Fong, Charles Thorpe, and Charles Baur. 2
- [10] Tom Vanderbilt and Brian Brenner. 2009. Traffic: Why We Drive the Way We Do (and What It Says about Us) , Alfred A. Knopf, New York, 2008; 978-0-307-26478-7. (2009).
- [11] Raja Parasuraman and Victor Riley. 1997. Humans and automation: Use, misuse, disuse, abuse. Human factors 39, 2 (1997), 230–253.
- [12] Thomas B Sheridan. 2002. Humans and automation: System design and research issues. Human Factors and Ergonomics Society.
- [13] ADAC: <https://www.adac.de/rund-ums-fahrzeug/ausstattung-technik-zubehoer/autonomes-fahren/grundlagen/autonomes-fahren-5-stufen/> (aufgerufen 17.11.2019)
- [14] Markus I. Reinke (2013): '30 Minuten Verkaufsprychologie'
- [15] <https://medium.com/punchcut/ux-design-for-autonomous-vehicles-9624c5a0a28f>
- [16] <https://link.springer.com/content/pdf/10.1007%2F978-0-85729-085-4.pdf>, Seiten 1271-1310
- [17] <https://iopscience.iop.org/article/10.1088/1757-899X/252/1/012096/pdf>
- [18] Bertoncello M and Wee D 2015 Ten ways autonomous driving could redefine the automotive world. McKinsey&Company <http://www.mckinsey.com/industries/automotive-and-assembly/our-insights/ten-ways-autonomous-driving-could-redefine-the-automotive-world>
- [19] <https://www.gruenderszene.de/lexikon/begriffe/usability?interstitial>
- [20] <https://www.usability.gov/what-and-why/user-centered-design.html>
- [21] https://www.seobility.net/de/wiki/User_Centered_Designs
- [22] Automatisierung - Von Fahrerassistenzsystemen zum automatisierten Fahren - VDA
- [23] https://www.focus.de/auto/autoentwicklung/bellbergs-auto-klartext-sorgen-autonome-autos-fuer-das-bedingungslose-grundeinkommen_id_10794360.html

- [24] <https://www.manager-magazin.de/unternehmen/autoindustrie/selbstfahrendes-kfz-mensch-vs-maschine-wer-gefahren-besser-erkennt-a-1200362.html>
- [25] <https://www.autonomes-fahren.de/musk-tesla-faehrt-autonom-zum-ende-des-jahres/>
- [26] <https://waymo.com/lidar/>
- [27] Christian Ress, Martin Wiecker - Potenzial der V2X-Kommunikation für Verkehrssicherheit und Effizienz; (im abstract)
- [28] Brian Reimer, PhD; MIT (ZITAT Youtube Video)
- [29] <https://t3n.de/news/tesla-statistik-unfaelle-autopilot-1136864/>
- [30] UPTANE - Security and Customizability of Software Updates for Vehicles; Trishank Karthik Kuppusamy, Lois Anne DeLong and Justin Cappos
- [31] https://www.youtube.com/watch?v=jTio_MPQRYc&t, Jeff Schneider Vortrag
- [32] <https://link.springer.com/content/pdf/10.1007%2Fs11623-017-0764-5.pdf>